Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek

Računarstvo usluga i analiza podataka

SEMINARSKI RAD

„Mushroom edibility editor“

Tena Kuzminski

Ana Kugler Burča

Osijek, 2025.

Sadržaj

1. Uvod

U projektnom zadatku prikazan je problem klasifikacije jestivosti šumskih gljiva na temelju njihovih svojstava. Motivacija prizlazi iz sve češćeg problema trovanja hranom zbog premale informiranosti o gljivama i njenim nuspojavama na zdravlje. Iako se često gljive nalaze u različitim receptima, nisu sve jestive iako su dostupne. Treba pripraziti pri kupovini, pohrani, načinu obrade te pripremanju gljiva.

Cilj ovog projektnog zadatka je olakšati prepoznavanje jestivih gljiva kroz jednostavnu aplikaciju.

2. Opis problema

Prilikom klasifikacije jestivosti gljiva promatraju se mnoga svojstva gljiva poput klobuka (*cap*), listića (*gill*), stručka (*stem*), opna (*veil*) i slično. Korištenjem pravilnih algoritama mogu se saznati ovisnosti podataka kako bi se predvidjela jestivost gljive. Kvalitetno razvijena i dizajnirana aplikacija omogućila bi različitim korisnicima informaciju o jestivosti gljiva na jednostavan način te sklonila sve sumnje u jestivost (moguće smetnje poput alergija, trovanja) pri korištenju gljiva. Sustav za klasifikaciju jestivosti šumskih gljiva o kojima govori projektni zadatak omogućava na jednostavan način ispitati, predvidjeti i analizirati jestivost gljive. Web aplikacija na vrlo jednostavan i intuitivan način, prikazuje jestivost gljiva nakon unosa nekoliko podataka koji su potrebni za precizno određivanje.

2.1. Korišteni podaci

Skup podataka koji se koristili za zadatak preuzet je sa stranice *UCI Machine Learning Repository* naziva *secondary+mushroom+dataset.csv*. Korišteni skup podataka sadrži 61069 hipotetskih gljiva sa klobucima na temelju 173 vrste (353 gljive po vrsti). Svaka gljiva je identificirana kao definitivno jestiva, definitivno otrovna ili nepoznate jestivosti i nije preporučljiva. Svaki zapis u skupu podataka obuhvaća sljedeće atribute:

- *cap-diameter* (promjer klobuka) – širina klobuka gljive, mjeri se u centimetrima
- *cap-shape* (oblik klobuka) – geometrijski obliku klobuka (npr.koničan, ravan, ispupčen)
- *cap-surface* (površina klobuka) - tekstura i izgled površine klobuka(glatka, ljuskava, sluzava)
- *cap-color* (boja klobuka) – dominantna boja klobuka
- *does-bruise-or-bleed* (mijenja li boju ili pušta sok) - reakcija mesa gljive na oštećenje – mijenja li boju ili pušta tekućinu
- *gill-attachment* (lamela, listići) – način na koji su listići (pločice ispod klobuka) pričvršćene na stručak
- *gill-spacing* (razmak listića) – udaljenost između listića, može biti gust, rijedak
- *gill-color* (boja listića) – boja, može pomoći u identifikaciji vrste
- *stem-height* (visina stručka) – duljina stučka, dijela koji nosi klobuk
- *stem-width* (debljina stručka) – promjer ili širina stručka
- *stem-root* (osnova stručka) – oblik ili prisutnost korijenastih struktura na dnu stručka
- *stem-surface* (površina stručka) – tekstura stručka, može biti glatka, vlaknasta, ljuskava
- *veil-type* (vrsta opne, zastorka) – vrsta opne koja pokriva ili štiti dijelove gljive tijekom razvoja (opći ili djelomični zastor)
- *veil-color* (boja opne) – boja membranskog sloja koji može ostati kao prsten ili ostatak na gljivi

- *ring-type* (vrsta prstena) – tip prstena koji ostaje oko stručka nakon pucanja opne
- *spore-print-color* (boja otiska spore) – boja koju ostavljaju spore na papiru – važna za identifikaciju
- *habitat* (stanište) – okruženje u kojem gljiva raste (šuma, livada, panj...)
- *season* (sezona) – vrijeme u godini kad se gljiva najčešće pojavljuje

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
class	cap-diameter	cap-shape	cap-surfac	cap-color	does-brui	gill-attach	gill-spacir	gill-color	stem-heig	stem-wid	stem-root	stem-surf	stem-colo	veil-type	veil-color	has-ring	ring-type	spore-prir	habitat	season
p	15.26	x	g	o	f	e		w	16.95	17.9.	s	y	w	u	w	t	g		d	w
p	16.6.	x	g	o	f	e		w	17.99	18.19	s	y	w	u	w	t	g		d	u
p	14.7.	x	g	o	f	e		w	17.8.	17.74	s	y	w	u	w	t	g		d	w
p	14.17	f	h	e	f	e		w	15.77	15.98	s	y	w	u	w	t	p		d	w
p	14.64	x	h	o	f	e		w	16.53	17.2.	s	y	w	u	w	t	p		d	w
p	15.34	x	g	o	f	e		w	17.84	18.79	s	y	w	u	w	t	p		d	u
p	14.85	f	h	o	f	e		w	17.71	16.89	s	y	w	u	w	t	g		d	w
p	14.86	x	h	e	f	e		w	17.3.	17.44	s	y	w	u	w	t	p		d	u
p	1.12.	f	g	o	f	e		w	17.27	18.69	s	y	w	u	w	t	p		d	a
p	13.55	f	g	e	f	e		w	16.4.	16.88	s	y	w	u	w	t	p		d	w
p	14.17	f	h	e	f	e		w	17.86	18.2.	s	y	w	u	w	t	p		d	a
p	13.4.	x	h	o	f	e		w	17.95	17.14	s	y	w	u	w	t	p		d	u
p	17.37	x	h	o	f	e		w	18.1.	18.27	s	y	w	u	w	t	g		d	u

Slika 1. Dio dataseta

Na podacima je odrađena deskriptivna statistika kojom je odrađeno opisivanje i sumiranje karakteristika podataka. Podaci su pripremljeni, organizirani, sažeti i omogućavaju prikaz na smisleni način kako bi lakše razumijeli njihova obilježja. Pomoću *data.info()* prikazani su podaci. Pojedini zapisi su imali nedostajuće vrijednosti (*data.isNull()*) koje su popunjene s dogovorenom vrijednosti *(u-unknown*) jer je imalo smisla dok su pojedini zapisi obrisani jer je bilo puno nedostajućih pa ne bi bili točni rezultati.

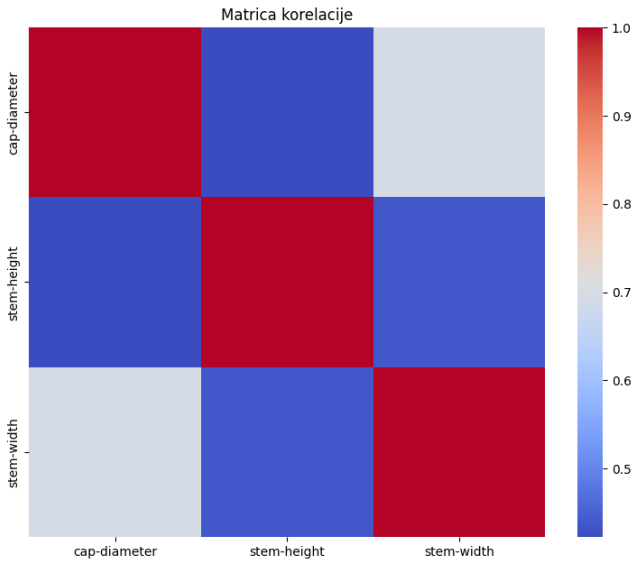
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 61069 entries, 0 to 61068
Data columns (total 21 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   class                 61069 non-null  object
 1   cap-diameter          61069 non-null  float64
 2   cap-shape             61069 non-null  object
 3   cap-surface           46949 non-null  object
 4   cap-color             61069 non-null  object
 5   does-bruise-or-bleed  61069 non-null  object
 6   gill-attachment       51185 non-null  object
 7   gill-spacing          36006 non-null  object
 8   gill-color            61069 non-null  object
 9   stem-height           61069 non-null  float64
 10  stem-width            61069 non-null  float64
 11  stem-root             9531 non-null   object
 12  stem-surface          22945 non-null  object
 13  stem-color            61069 non-null  object
 14  veil-type             3177 non-null   object
 15  veil-color            7413 non-null   object
 16  has-ring              61069 non-null  object
 17  ring-type             58598 non-null  object
 18  spore-print-color     6354 non-null   object
 19  habitat               61069 non-null  object
 20  season                61069 non-null  object
dtypes: float64(3), object(18)
memory usage: 9.8+ MB
```

Slika 2. Prikaz *data.info()*

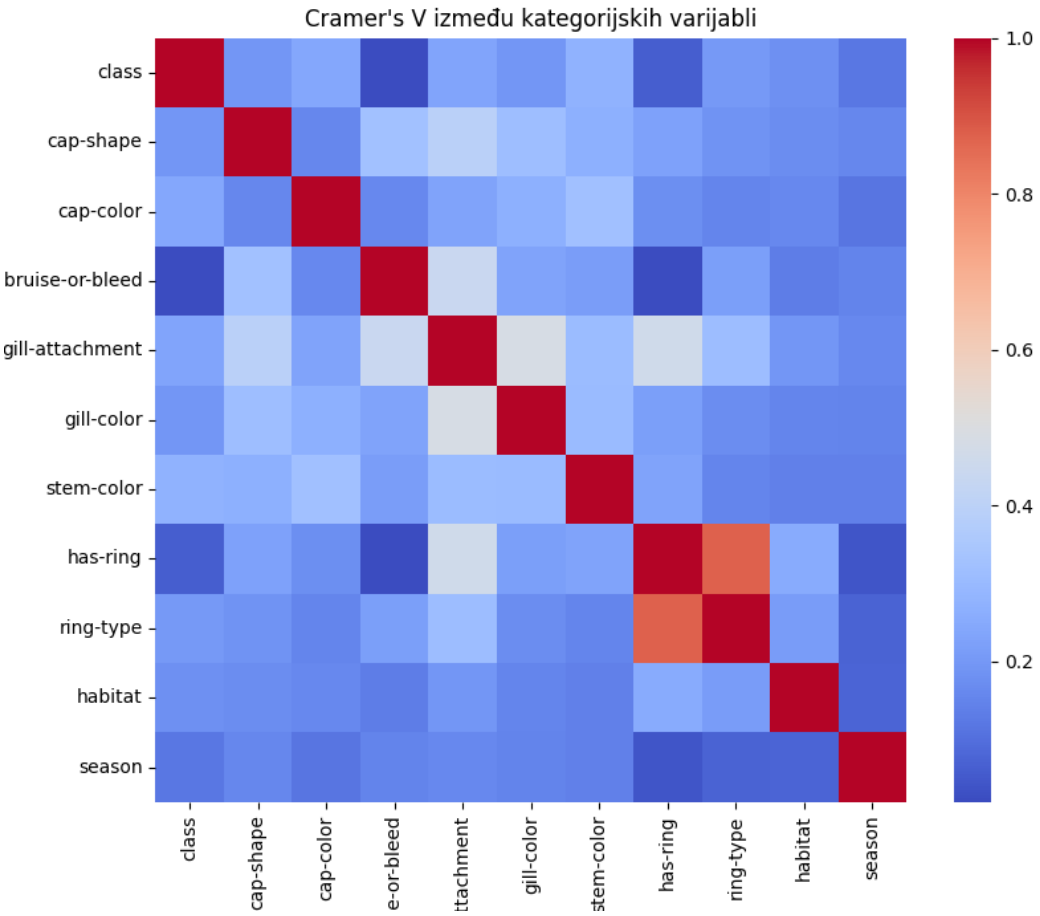
```
class                     0
cap-diameter              0
cap-shape                 0
cap-surface           14120
cap-color                 0
does-bruise-or-bleed      0
gill-attachment        9884
gill-spacing          25063
gill-color                0
stem-height               0
stem-width                0
stem-root             51538
stem-surface          38124
stem-color                0
veil-type             57892
veil-color            53656
has-ring                  0
ring-type              2471
spore-print-color     54715
habitat                   0
season                    0
dtype: int64
```

Slika 3. Nedostajuće vrijednosti *isNull()*

Korelacijska matrica izrađena je pomoću naredbe *corr_matrix* za numeričke varijable kako bi se prikazala povezanost između podataka što označava vrijednost jedne varijable s nekom vjerojatnošću moguće predvidjeti na osnovi saznanja o vrijednosti druge varijable. Promjena vrijednosti jedne varijable utječe na promjenu vrijednosti druge varijable. Za kategorijske varijable korištena je *cramers_v(conf_matrix).* Ni jedne dvije varijable nisu jako korelirale te su zbog toga ostavljene sve varijable.



Slika 4. Matrica korelacije za numeričke varijable (*cap-diameter, stem-height, stem-width*)



Slika 5. Matrica korelacije za kategorijske varijable (*class, cap-shape, cap-color, bruise-or-bleed, gill-attachment, gill-color, stem-clolor, has-ring, ring-type, habitat, season*)

2.2. Korišteni postupci strojnog učenja

Nakon pripremljenih i pročišćenih podataka, za izradu modela koji će predviđati jestivosti šumskih gljiva, potrebno je koristiti metode za odvajanje podataka na trening i test, odabrati algoritam za strojno učenje, usporediti stvarne i predviđene vrijednosti te ocijeniti model predviđanja. Podaci se odvajaju na trening i test kako bi se model na temelju postavljenih pravila i obrađenih podataka mogao istrenirati za predviđanje vrijednosti. Tijekom razvoja modela testirano je više algoritama strojnog učenja kako bi se pronašlo najbolje rješenje za zadane podatke jer nije svaki algoritam pogodan za sve tipove podataka.

Model	Brzina	Točnost	Tumačivost	Otporan na prenaučenost
Decision Tree	✅ Brz	⚠️ Umjerena	✅ Visoka	❌ Ne
Random Forest	⚠️ Sporiji	✅ Visoka	⚠️ Srednja	✅ Da
kNN	❌ Spor (predikcija)	⚠️ Umjerena	✅ Visoka	❌ Ne
SVC	⚠️ Sporiji	✅ Visoka	⚠️ Srednja	✅ Da
Gradient Boosting	❌ Spor	✅✅ Vrlo visoka	⚠️ Srednja	⚠️ Ovisi o podešavanju

Slika 6. Usporedni rezultati korištenih algoritama

Algoritmi za strojno učenje:

1. Stablo odluke - *"Decision Tree": DecisionTreeClassifier(random_state=42)*

Stablo odluke je algoritam nadziranog učenja koji se koristi za klasifikacijske i regresijske zadatke, a modelira odluke na temelju ulaznih značajki, stvarajući strukturu nalik stablu. To je alat za podršku odlučivanju koji vizualno prikazuje moguće ishode, posljedice i troškove određene odluke, pomažući u procjeni i usporedbi različitih opcija djelovanja.

AUC- Area under the Curve – Površina ispod krivulje, najčešće koristi ROC krivulju (Reciever Operating Characteristic). Pokazuje koliko je model dobar u razdvajanju klasa (npr.pozitivno vs. negativno). *Mean AUC* – Prosječni *AUC*- predstavlja srednju vrijednost svih rezultata kod križnih provjera (cross-validation) ili ponavljanja što označava pokazatelja opće točnosti modela.

```
Tuning Decision Tree...
 Params: {'max_depth': 3}, Mean AUC: 0.7268
 Params: {'max_depth': 5}, Mean AUC: 0.8044
 Params: {'max_depth': 10}, Mean AUC: 0.9355
```

Slika 7. *Decision Tree* parametri

Koristeći algoritam *Decision Tree* može se vidjeti da povećanjem max_depth (maksimalne dubine stabla) raste *AUC*. Dublje stablo uči složenije obrasce te ima bolju predikciju. Treba pripaziti da ne dođe do overfittinga ako se previše produbi.

2. Šuma stabala - *"Random Forest": RandomForestClassifier(random_state=42),*

Random Forest - Slučajna šuma stabala je algoritam strojnog učenja koji koristi ansambl stabala odluke za izvođenje predikcija. Kombinira izlaze više stabala odluke kako bi se postigao robusniji i točniji rezultat, što ga čini prikladnim za klasifikacijske i regresijske zadatke.

```
Tuning Random Forest...
 Params: {'n_estimators': 50, 'max_depth': 5}, Mean AUC: 0.9272
 Params: {'n_estimators': 50, 'max_depth': 10}, Mean AUC: 0.9970
 Params: {'n_estimators': 100, 'max_depth': 5}, Mean AUC: 0.9332
 Params: {'n_estimators': 100, 'max_depth': 10}, Mean AUC: 0.9974
```

Slika 8. *Random Forest* parametri

Koristeći algoritam *Decision Tree* može se vidjeti da kombinacijom više stabala (npr.100 umjesto 50) poboljšava se stabilnost modela te da veća dubina donosi bolje rezultate. Kombinacijom *max_depth=10* i *n_estimators=100* daje skoro savršen *AUC* što prikazuje da je to jako dobar model.

3. Algoritam k-najbližih susjeda - *"kNN": KNeighborsClassifier(),*

Algoritam k-najbližih susjeda (*kNN*) je neparametarski, nadzirani klasifikacijski algoritam koji koristi blizinu (udaljenost) za donošenje klasifikacija ili predikcija o pripadnosti pojedinog podatkovnog uzorka nekoj skupini.

```
Tuning kNN...
 Params: {'n_neighbors': 3}, Mean AUC: 0.9999
 Params: {'n_neighbors': 5}, Mean AUC: 0.9999
 Params: {'n_neighbors': 7}, Mean AUC: 0.9999
```

Slika 9. *kNN* parametri

Svi rezultati su izvanredno visoki (0.9999) što prikazuje da broj n susjeda ne pravi značajnu razliku.

4. Klasifikator pomoću potpornih vektora - *"SVC": SVC(probability=True, random_state=42),*

SVC (klasifikator pomoću potpornih vektora) je posebna implementacija algoritma potpornih vektora *(SVM)*, namijenjena isključivo klasifikacijskim zadacima. Drugim riječima, *SVC* je varijanta *SVM*-a koja se koristi za klasifikaciju. Cilj mu je pronaći hiperplohu koja najbolje razdvaja podatkovne točke u različite klase.

```
Tuning SVC...
 Params: {'C': 0.1, 'kernel': 'rbf'}, Mean AUC: 0.9105
 Params: {'C': 1, 'kernel': 'rbf'}, Mean AUC: 0.9871
 Params: {'C': 10, 'kernel': 'rbf'}, Mean AUC: 0.9998
```

Slika 10. *SVC* parametri

Povećavanjem vrijednosti c omogućeno je modelu da jače „kažnjava“ pogreške, što vodi ka složenijem modelu. Perfomanse poprilično rastu sa c=10.

5. Gradijentno poboljšavanje - *"Gradient Boosting" :GradientBoostingClassifier(random_state=42),*

Gradijentno poboljšavanje (*Gradient Boosting*) je snažna tehnika strojnog učenja koja se koristi za regresijske i klasifikacijske zadatke. Radi tako da postupno kombinira više slabih učenika (najčešće stabala odluke) kako bi se izgradio snažan prediktivni model. Algoritam iterativno nadograđuje prethodne modele, usmjeravajući se na pogreške koje su ti modeli napravili, s ciljem poboljšanja ukupne točnosti.

```
Tuning Gradient Boosting...
 Params: {'n_estimators': 50, 'max_depth': 5}, Mean AUC: 0.9956
 Params: {'n_estimators': 50, 'max_depth': 10}, Mean AUC: 1.0000
 Params: {'n_estimators': 100, 'max_depth': 5}, Mean AUC: 0.9994
 Params: {'n_estimators': 100, 'max_depth': 10}, Mean AUC: 1.0000
```

Slika 11. *Gradient Boosting* parametri

Daje najbolje rezultate sa *max_depth=10 – AUC= 1.0000*. Vrlo moćan algoritam, samo se mora pravilno podesiti i pripaziti da se ne *overfita.*

Najbolji algoritmi po *AUC* je *Gradient Boosting* koji je vrlo jako algoritam za tabularne podatke, uči postepeno te svaki novi model pokušava ispraviti pogrešku prehodnog modela. Prvi model napravi osnovnu predikciju dok sljedeći model uči rezidue(greške) tog modela i pri tome ispravlja pogreške prethodnog. Na kraju svi modeli se kombiniraju i naprave konačnu predikciju.

```
Najbolji parametri i AUC po modelu:
Decision Tree: {'max_depth': 10}, AUC: 0.9355
Random Forest: {'n_estimators': 100, 'max_depth': 10}, AUC: 0.9974
kNN: {'n_neighbors': 5}, AUC: 0.9999
SVC: {'C': 10, 'kernel': 'rbf'}, AUC: 0.9998
Gradient Boosting: {'n_estimators': 100, 'max_depth': 10}, AUC: 1.0000
```

Slika 12. Najbolji parametri i *AUC* po modelu

Na temelju rezultata najbolji algoritam je *Gradient Boosting*.

3. Opis programskog rješenja

Programsko rješenje projektnog zadatka sastoji se od nekoliko komponenti:

- Model strojnog učenja – spremljen u .pkl formatu koristeći biblioteku joblib
- Flask API – prima korisničke zahtjeve i vraća predviđene rezultate
- Web sučelje – unos korisničkih podataka i prikaz predviđanja modela
- Deploy rješenje putem platforme Render.com

3.1. Model strojnog učenja

Google Colab korišten je za izradu i treniranje modela. Odabran je za korištenje zbog svoje jednostavnosti pri radu, lakšeg upravljanja resursima,otkrivanju i otklanjanju pogrešaka tijekom rada. Velika prednost je dostupnost svima, besplatna verzija koja nema vremenskih ograničenja, ograničenja resursa, dodatnih nadoplata i sličnih problema koji su postojali s Microsoft Azure.

Model je treniran za klasifikacije jestivosti šumskih gljiva na temelju njihovih svojstava. Za izgradnju modela korišten Gradient Boosting algoritam zbog svoje odlične ukupne točnosti. U usporedbi s drugim algoritmima koje smo koristili prednost kod njega je što koristi uči postepeno i ispravlja greške prethodnog modela. Na kraju se svi modeli kombiniraju i prave konačnu predikciju velike točnosti.

Učitavanje korištenih biblioteka

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import scipy.stats as stats
import numpy as np
from statsmodels.stats.multicomp import pairwise_tukeyhsd
from sklearn.preprocessing import LabelEncoder
import statsmodels.api as sm
from scipy.stats import chisquare, chi2_contingency
```

```
from sklearn.model_selection import StratifiedKFold, train_test_split, cross_validate
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier, GradientBoostingClassifier
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.metrics import (
    accuracy_score,
    f1_score,
    recall_score,
    precision_score,
    roc_auc_score,
    confusion_matrix,
    classification_report
)
```

```
import mlflow
import mlflow.sklearn
import joblib
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

Učitavanje i prikaz podataka

```
data = pd.read_csv('secondary_data.csv', sep=';')


data.info()
```

Podjela na trening i test skup:

Jedan od najčešće korištenih pristupa za ovu svrhu je križna validacija (cross-validation). Križna validacija omogućava da se cijeli dostupni skup podataka višestruko iskoristi i za treniranje i za testiranje, čime se smanjuje rizik od preprilagodbe (*overfittinga*) te se dobiva robusnija mjera performansi modela.

Koristi se *StratifiedKFold(n_splits=5),* koji označava da se podaci dijele na 5 jednakih dijelova (20% svaki) te se u svakoj od 5 iteracija 1/5 podataka (20%) koristi se kao testni skup dok se 4/5 podataka (80%) koristi se kao trenirajući skup. Nakon jedne iteracije, *foldovi* se rotiraju – svaki podatak će jednom biti u testnom skupu, a četiri puta u trenirajućem. Tako se dobije prosječnu procjenu performansi modela na temelju 5 različitih podjela.

```
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
```

```
for train_idx, test_idx in skf.split(X, y):
    X_train_cv, X_test_cv = X.iloc[train_idx], X.iloc[test_idx]
    y_train_cv, y_test_cv = y.iloc[train_idx], y.iloc[test_idx]
```

Treniranje modela:

```
model.fit(X_train_cv, y_train_cv)
```

Evaluacija modela:

Evaluacijske metrike:

- *Accuracy* – točnost – mjeri omjer točno predviđenih primjera u odnosu na ukupan broj primjera.
- *Precision* – preciznost – mjeri koliko od svih pozitivnih predikcija modela stvarno je pozitivno prepoznao.

- *F1* – f1 mjera – harmonijska sredina između preciznosti i odziva.
- *Roc_auc* – *Auc* pod *ROC* krivuljom – sposobnost modela da razlikuje klase (pozitivnu i negativnu.

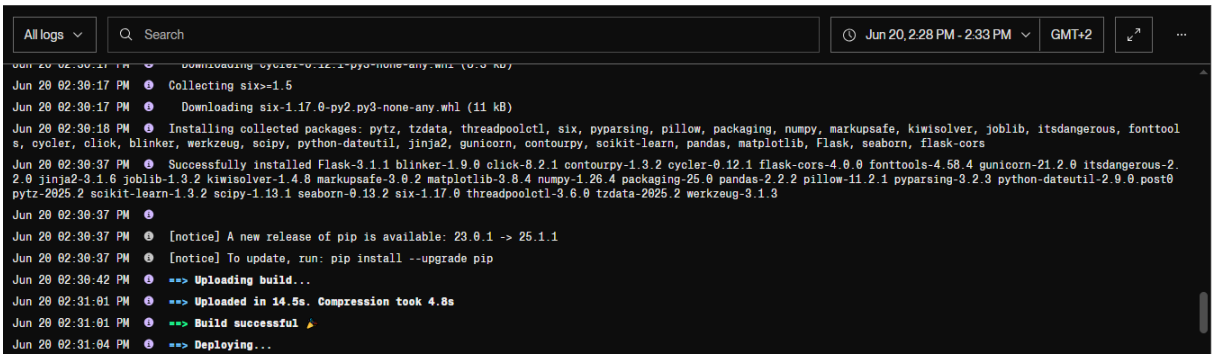
```python
# Evaluacija svih modela s najboljim parametrima preko 5-fold CV i spremanje rezultata za vizualizaciju
scoring = {
    'accuracy': 'accuracy',
    'f1': 'f1',
    'recall': 'recall',
    'precision': 'precision',
    'roc_auc': 'roc_auc'
}

results = {}

print("\n=== Evaluacija svih modela s najboljim parametrima (5-fold CV) ===\n")

for name, base_model in models.items():
    print(f"Evaluacija modela: {name}")
    model = base_model.__class__(**best_params[name])
    pipeline = Pipeline([
        ('scaler', StandardScaler()),
        ('classifier', model)
    ])
    cv_result = cross_validate(pipeline, X, y, cv=skf, scoring=scoring, return_train_score=False)
    results[name] = cv_result
    for metric in scoring.keys():
        scores = cv_result[f'test_{metric}']
        print(f"  {metric}: {scores.mean():.4f} ± {scores.std():.4f}")
    print()
```

```
Evaluacija modela: Decision Tree
  accuracy: 0.8487 ± 0.0109
  f1: 0.8602 ± 0.0082
  recall: 0.8452 ± 0.1117
  precision: 0.9029 ± 0.1005
  roc_auc: 0.9355 ± 0.0045

Evaluacija modela: Random Forest
  accuracy: 0.9747 ± 0.0014
  f1: 0.9772 ± 0.0012
  recall: 0.9780 ± 0.0021
  precision: 0.9764 ± 0.0036
  roc_auc: 0.9976 ± 0.0003

Evaluacija modela: kNN
  accuracy: 0.9999 ± 0.0001
  f1: 0.9999 ± 0.0001
  recall: 0.9999 ± 0.0001
  precision: 0.9999 ± 0.0001
  roc_auc: 1.0000 ± 0.0000

Evaluacija modela: SVC
  accuracy: 0.9997 ± 0.0002
  f1: 0.9997 ± 0.0002
  recall: 1.0000 ± 0.0000
  precision: 0.9995 ± 0.0003
  roc_auc: 1.0000 ± 0.0000

Evaluacija modela: Gradient Boosting
  accuracy: 0.9999 ± 0.0001
  f1: 0.9999 ± 0.0001
  recall: 0.9999 ± 0.0001
  precision: 0.9999 ± 0.0001
  roc_auc: 1.0000 ± 0.0000
```

Slika 13. Evaluacije modela

Spremanje treniranog modela

```
model_file = f'mushroom_model_pipeline_{best_model_name}.pkl'
joblib.dump(best_model_pipeline, model_file)
print(f"Najbolji model spremljen kao pipeline u {model_file}")
```



Slika 14. Prikaz buildanja modela deploy

3.2. Način korištenja API-ja

Aplikacijsko programsko sučelje *(API)* realizirano je pomoću *Flask* okvira u programskom jeziku *Python*. *API* omogućuje komunikaciju između korisničkog sučelja i modela strojnog učenja, te je osmišljen kako bi primao podatke, obrađivao ih kroz trenirani model i vraćao rezultat predviđanja.

API definira dvije osnovne rute:

– "/" – vraća jednostavnu poruku koja potvrđuje da je *API* aktivan i dostupan,

– "/predict" – prima *JSON* objekt koji sadrži šest značajki (ulaznih vrijednosti) te vraća

predviđenu vrijednost rasta prihoda.

Učitavanje modela:

```
import joblib
model = joblib.load("mushroom_model_Gradient Boosting.pkl")
```

Osnovna struktura *Flask* aplikacije:

```
from flask import Flask, request, jsonify
from flask_cors import CORS
app = Flask(__name__)
CORS(app)
```

Ruta koja potvrđuje da je *API* aktivan:

```
@app.route('/')

def home(): return

"API radi. Pošalji POST na /predict."
```

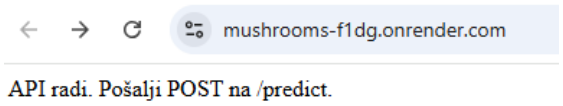
Ruta koja prima podatke i vraća predviđanje:

```
@app.route('/predict', methods=['POST'])

def predict(): data = request.get_json() features = data.get("features",
[]) prediction = model.predict([features]) return jsonify({"prediction":
prediction[0]})
```

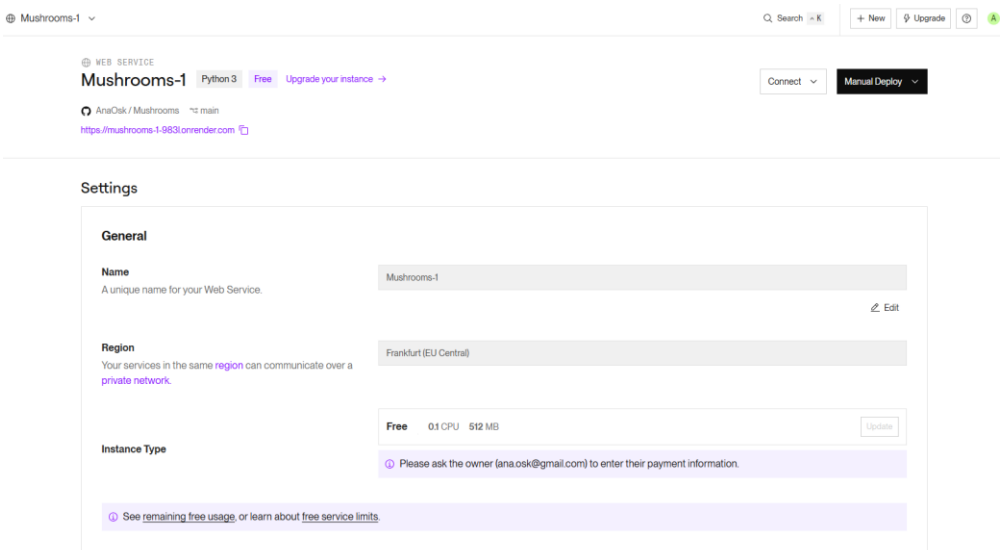
Model *i API* su postavljeni na Render.com, što je omogućilo besplatno hostanje aplikacije.

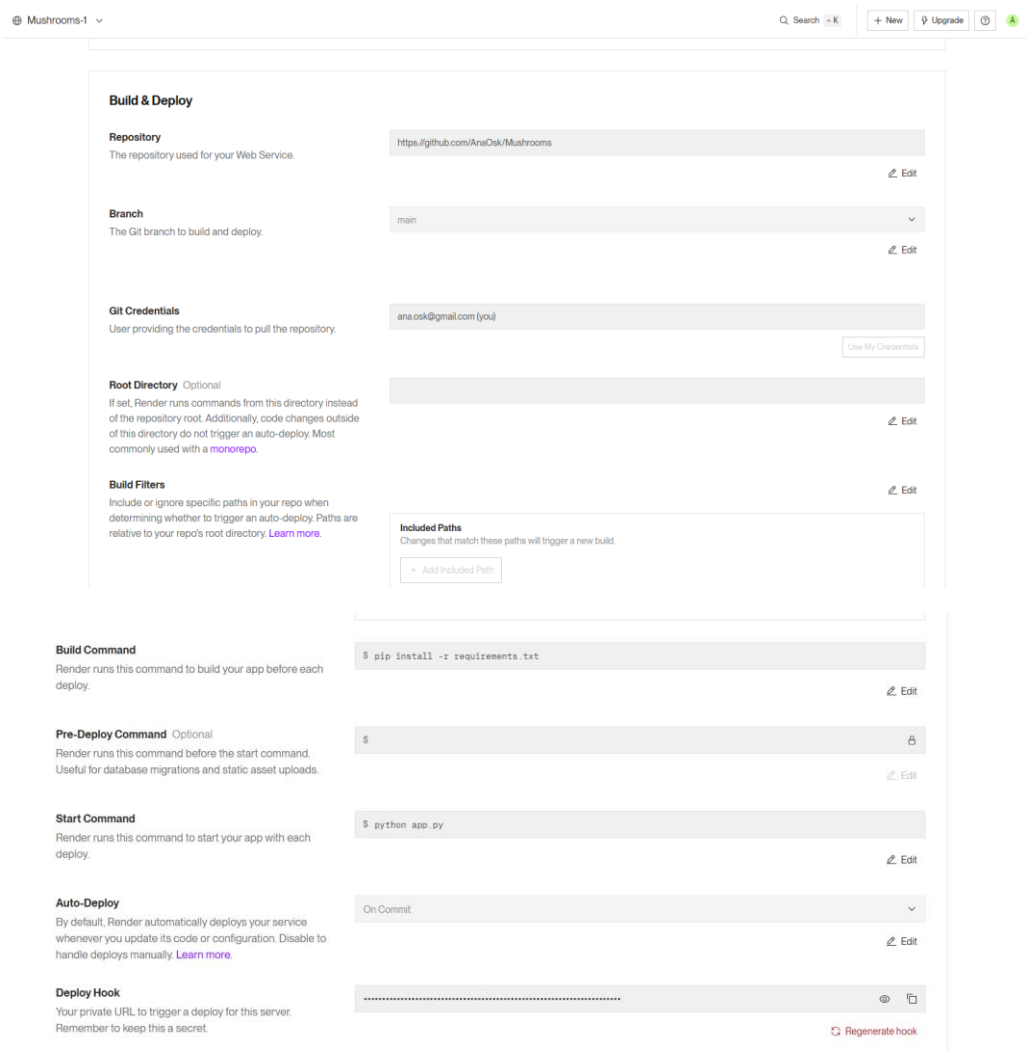
Pomoću *CORS (Cross-Origin Resource Sharing)* omogućeno je da *HTML* forma iz drugog izvora može komunicirati s *API-jem*. Kada korisnik unose podatke, forma ih šalje kao *JSON*, a odgovor modela odnosno predviđanje, se prikazuje u stvarnom vremenu.

Online API adresa: https://mushrooms-f1dg.onrender.com/



Slika 15. API prikaz

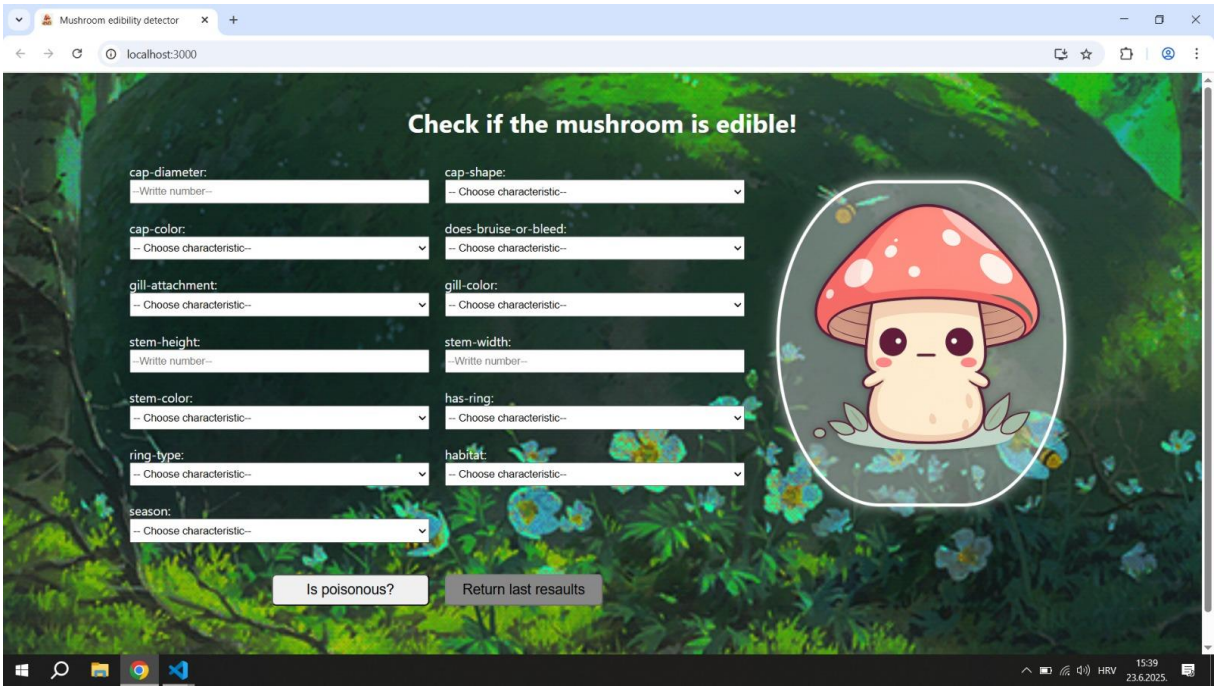




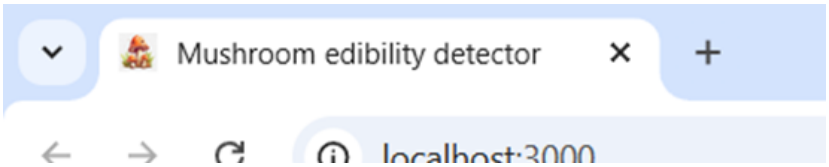
Slika 16. Render setup

3.3. Klijentska aplikacija

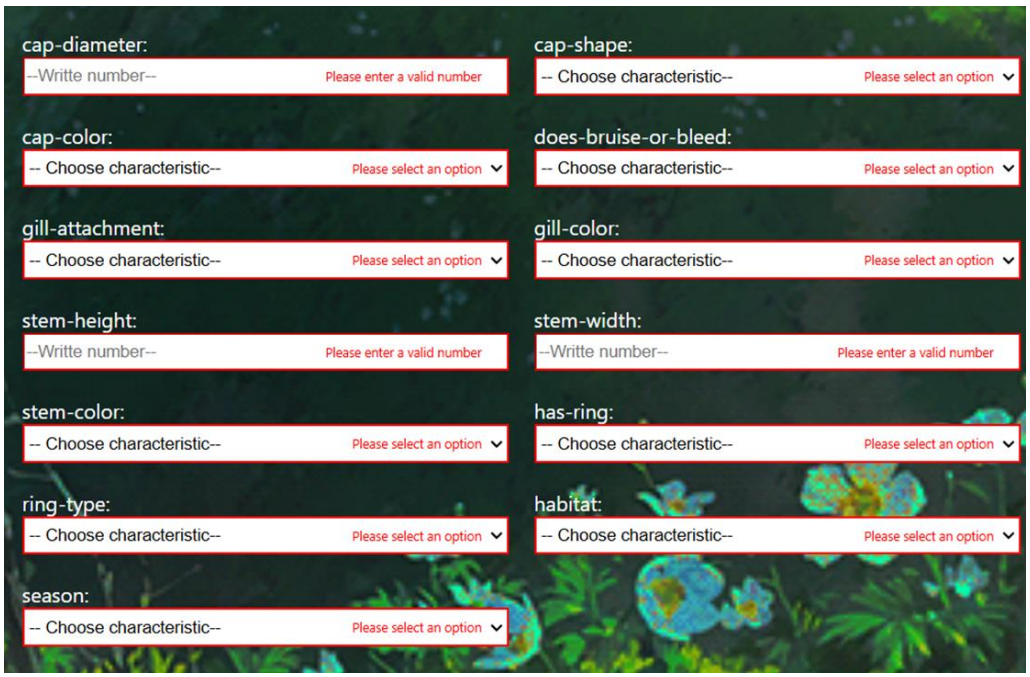
Web aplikacija je jednostavno i intuitivno sučelje koji traži upis nekoliko podataka. Potrebno je popuniti sve podatke kako bi model mogao prema zadani podacima odrediti jel gljiva čije smo značajke unijeli jestiva ili nije. Osim tekstualnog unosa sljedeći podataka: *cap-diameter, cap-color, cap-shape, does-bruise-or-bleed, gill-attachment, gill-color, stem-height, stem-width, stem-color, has-ring, ring-type, habitat, seaseon* , gumba za spremanje upisa i odluke kakva je gljivas, s desne strane imamo vizualni prikaz gljive prema kojem možemo prepoznati osobine gljive. U svakom polju za unos podataka dodan je *Hint tekst* koji opisuje korisniku što treba unijeti i da je polje obavezno za daljnu interakciju. Nije potrebno registrirati se ili ulogirati u aplikaciju. Moguće je još vidjeti prethodni rezultat pritiskom na gumb *Return last resaults* .



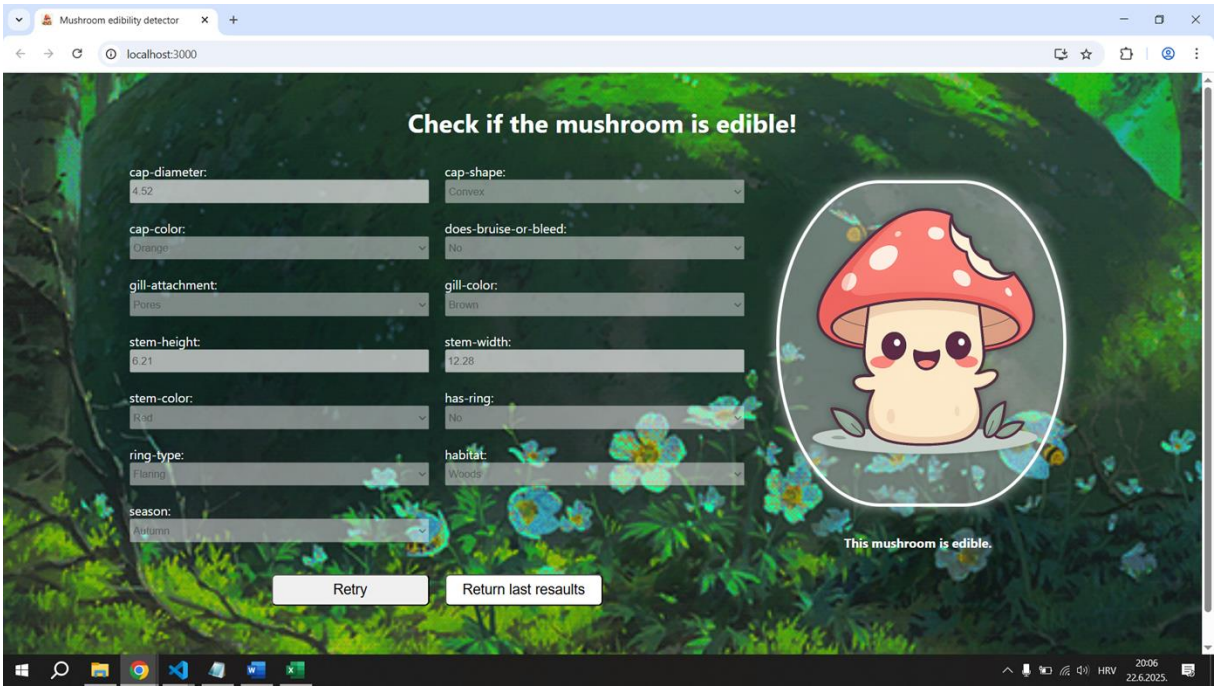
Slika 17. Početni screen



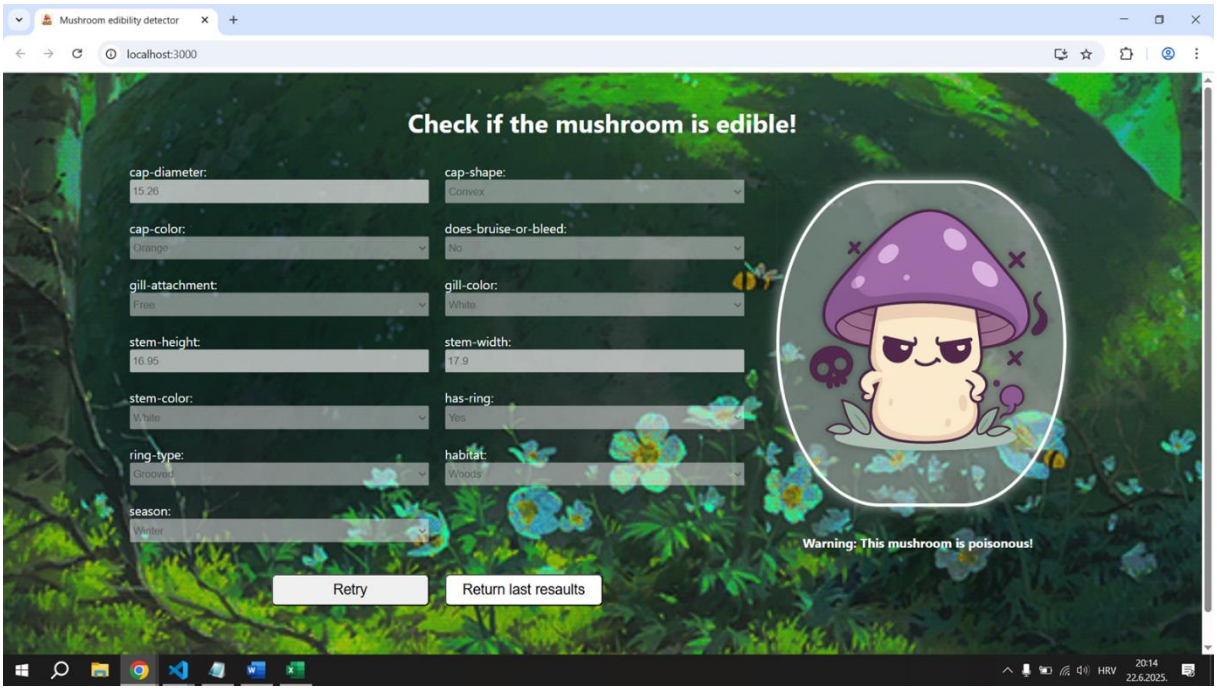
Slika 18. Prikaz naziva i ikone u tabu preglednika



Slika 19. Prikaz dok nisu upisani podaci



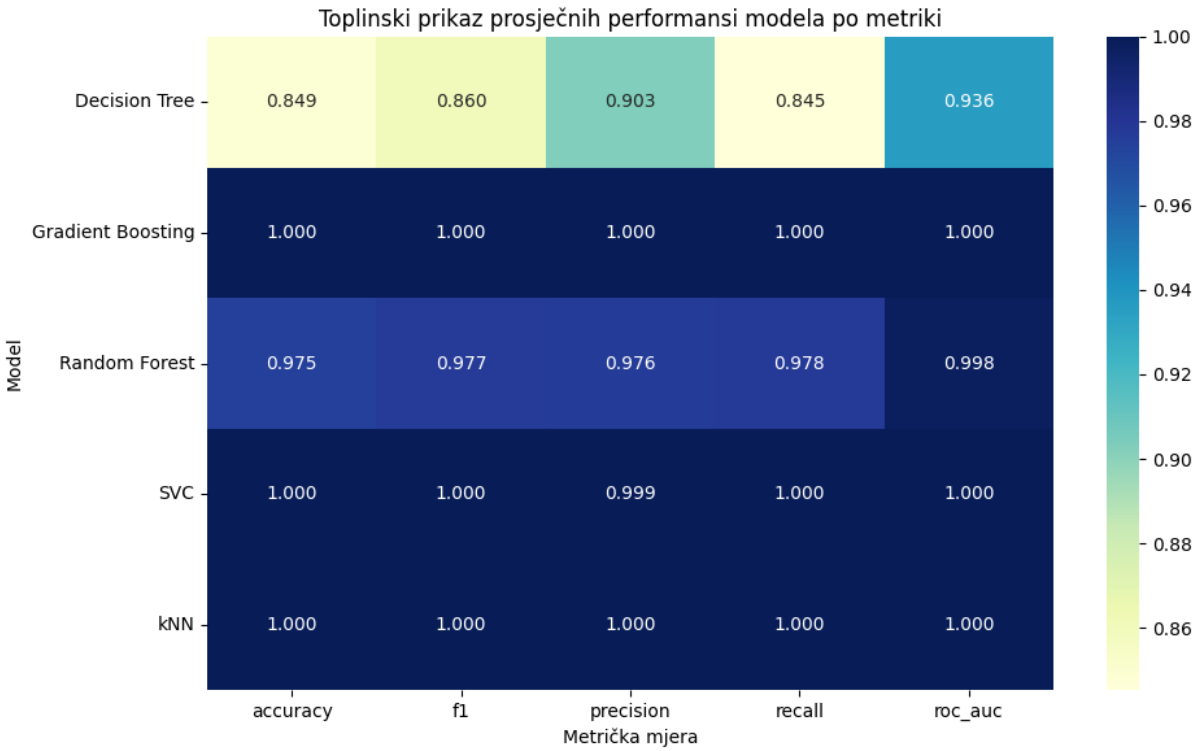
Slika 20. Unešeni podaci – jestiva gljiva



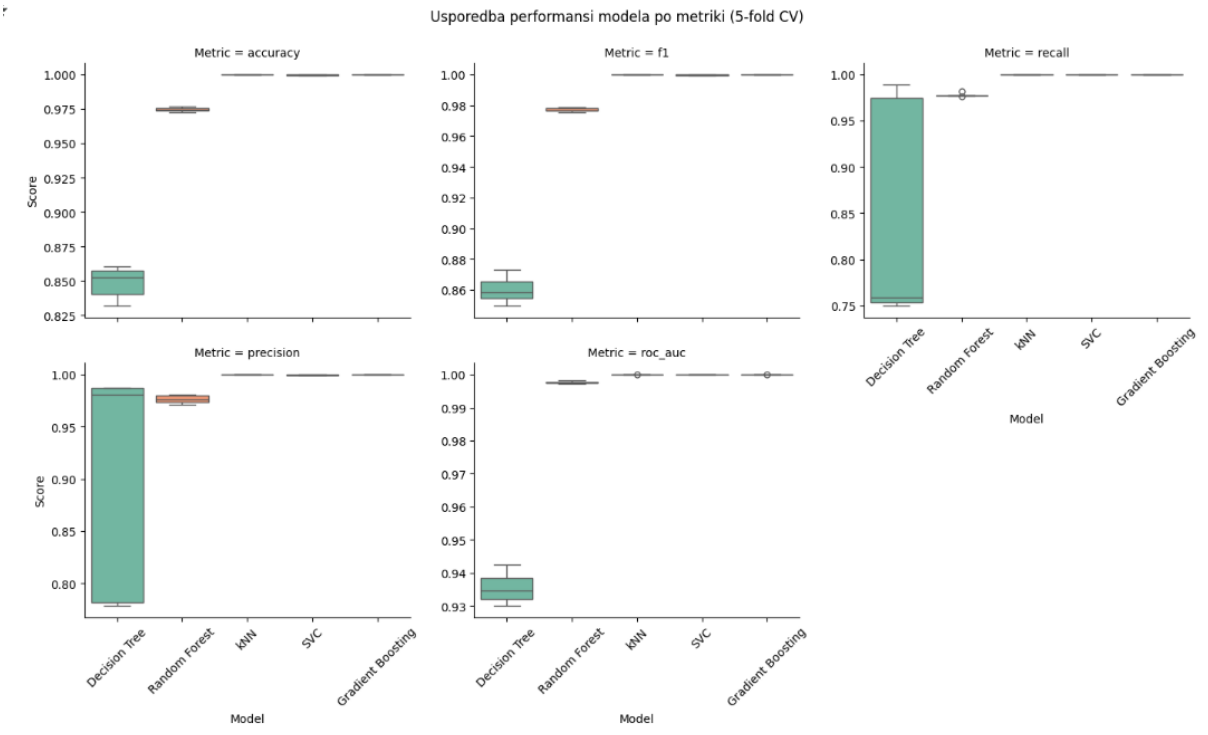
Slika 21. Unešeni podaci – otrovna gljiva

3.4. Dodatno

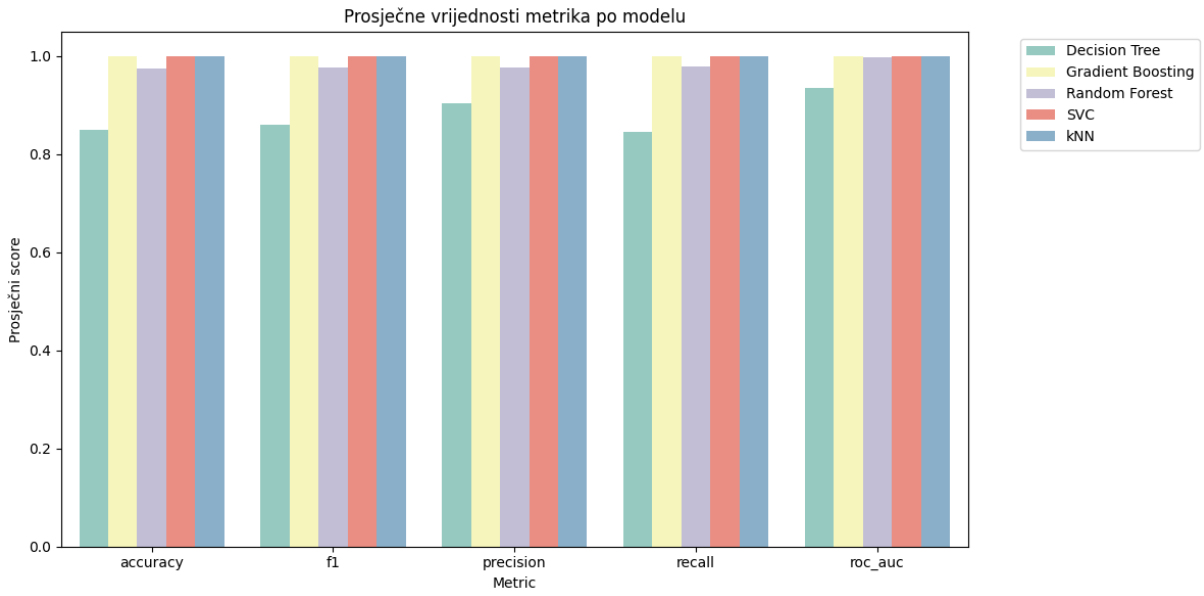
Vizualizacije



Slika 22. Toplinski prikaz prosječnih performansi modela po metriki



Slika 23. Usporedba performansi modela po metriki (5-fold CV)



Slika 24. Prosječne vrijednosti metrika po modelu

4. Zaključak

U ovom projektnom zadatku razvijen je sustav za rješavanje problema klasifikacije jestivosti šumskih gljiva na temelju njihovih svojstava u obliku jednostavne web aplikacije. Web aplikacija naziva Mushroom edibility editor je vrlo jednostavna i *user-friendly* s nekoliko funkcionalnosti. Korisnicima omogućava unos nekoliko osobina gljiva prema kojima dobiju informaciju jel određena gljiva jestiva ili ne.

Model strojnog učenja treniran je u *Google Colab* okruženju korištenjem algoritma *Gradient Boosting* prema kojem smo dobili najbolje rezultate. *Web API* sučelje temeljeno je na *Flasku* te omogućuje unos podataka i dobivanje odgovora o jestivosti gljive. Cijelo rješenje postavljeno je na platformu *Render.com*

Svojom brzinom rada i jednostavnošću korištenja aplikacija *Mushroom edibility editor* zajedno s istreniranim modelom predstavlja savršeno rješenje za šumske ljude koji žele uživati u okusima gljiva bez posljedica.

5. Poveznice i literatura

Programskom je rješenju moguće pristupiti preko:

Programsko rješenje na GitHubu
ML model
Secondary mushroom dataset