Univerzitet u Kragujevcu

SEMINARSKI RAD

iz predmeta Mašinsko učenje

Tema:

Klasifikacija lokalizacije strana proteina

Mentor : Ognjen Pavić
Student : Željko Simić 3vi/2023

Kragujevac 2024.

1. Uvod

Dat je 'Yeast' (kvasac)[1] skup podataka sadrži 1484 instanci sa 8 neklasifikatornih atributa kontinualnih vrednosti za svaku instancu (ne gleda se 1. neklasifikatorni atribut koji je kategorički). Ideja je da se izvši ponovna obuka nad više raznolikih modela algoritama, tako što će se neklasnim atributima obučavati uz željene vrednosti po 10 kategoričkih vrednosti klasifikatornog atributa [' MIT', ' NUC', ' CYT', ' ME1', ' EXC', ' ME2', ' ME3', ' VAC', ' POX', ' ERL'] (dobijenih implementacijom 1. u Klasifikacije.prikaz_klasifikatornih_vrednosti()). A kasnije koristiti za evaluaciju predikcija uzoraka naspram kojih će se dobijati te klasifikatorne vrednosti kao izlaz naspram ulaza sastavljenog od vrednosti neklasifikatornih atributa. Ovde su pobrojani i objašnjeni neklasifikatorni atributi (sa vrednostima unapred sračunatim i bez nedodostajućih vrednosti):

1. **Sequence_Name**: Pristupna šifra za SWISS-PROT bazu podataka, kategorička vrednost. Jednoznačan je, a i nije toliko značajan. Pogodno je ne koristiti ga.
2. **mcg**: McGeoch-ov metod za prepoznavanje signalnih sekvenci.
3. **gvh**: Von Heijne-ov metod za prepoznavanje signalnih sekvenci.
4. **alm**: Ocena po ALOM plazma-membranskoj rasprostranjenosti programa predviđanja.
5. **mit**: Ocena po analizi disktriminatornih vrednosti nad sadržajima amino kiselina u N-terminal sredinama (20. stepena po ostatku) mitohondričnih i nemitohondričnih proteina.
6. **erl**: Zastupljenost HDEL podniske (kroz delovanje kao signal za ponovno zauzimanje u endoplazmičnom lumena retikuluma).
7. **pox**: Peroksisomalno ciljajuće signaliziranje u C-terminusu.
8. **vac**: Ocena analize diskriminantih vrednosti sadržaja amino kiseline vakuolarnih i ekstraćelijskih proteina.
9. **nuc**: Ocena analiza diskriminantnih vrednosti signala jezgarnih lokalizacija jezgarnih i nejezgarnih proteina.

Klasifikatorni atribut koji će biti posmatran je:

1. **localization_site:** Lokalizaciju strana ćelijskih proteina.

2. Metodologija

Vrši se učitavanje zadatog skupa podataka, procesiranje podataka u oblik pogodan za korišćenje algoritama mašinskog učenja i podela skupa na trening i test deo u implementaciji 1. Zarad neke prividno bolje organizacije koristi se koncept klasa objektno-orjentisanog programiranja zarad definisanja pojedinih procedura oko polja data tipa pandas.DataFrame. Definišu se metode za enkodiranje vrednosti 1. neklasifikatornog kategoričkog atributa i klasifikatornog atributa u neki enumerisani, konstruktor OOP klase. Navedeno 'yeast/yeast.csv' je relativna putanja do skupa podataka koji je izmenjena verzija preuzetog yeast.data datoteke. Vrši se OOP instanciranje klase u promenljivu klasifikacije. Dalje je vršena podela na skup podataka namenjenog za obuku i testiranje, po navođenju feature(neklasifikatornih) atributa, a i klasifikatornih sa navedenom veličinom skupa podataka testiranja, za sada, od 20%.

```
import pandas as pd
from sklearn.model_selection import train_test_split
# ...

class Klasifikacije:

	def __init__(self, data):
		self.data = data

		# uklonim duplikate
		self.data.drop_duplicates()
		self.uradi_enkodiranje_prvog_atributa()
		self.uradi_enkodiranje_klasnog_atributa()

	# koristim klase zato što je jednostavnije za rad nad dataset promenljivom
	def uradi_enkodiranje_prvog_atributa(self):
		temp = dict()
		for i, value in enumerate(self.data[self.data.columns[0]]):
			temp[value] = i
		self.data[self.data.columns[0]] = self.data[self.data.columns[0]].map(temp)
	def uradi_enkodiranje_klasnog_atributa(self):
		temp = dict()
		for i, value in enumerate(self.data[self.data.columns[-1]].unique()):
			temp[value] = i
		self.data[self.data.columns[-1]] = self.data[self.data.columns[-1]].map(temp)
	def prikaz_klasifikatornih_vrednosti(self):
		print('Vrednosti klasifikatornog atributa')
		print(list(self.data[self.data.columns[-1]].unique()))


input_data = pd.read_csv('yeast/yeast.csv')
klasifikacije = Klasifikacije(input_data)


x_train, x_test, y_train, y_test = train_test_split(
	klasifikacije.data[klasifikacije.data.columns[:-1]],
	klasifikacije.data[klasifikacije.data.columns[-1]],
	test_size=0.2
	)
```

Implementacija 1: Priprema funkcija za procesiranje i prezentovanje podataka

Kreiranje nekoliko modela mašinskog učenja klasifikacije i njihovo medjusobno poređenje vrši se, za sada, nad raznovrsnim algoritmima naivnog Bajesa, prikazanog u implementaciji 2. Svaki model će se prilagoditi obukom sa metodom .fit(...) nad skupom podataka namenjenog za obuku vrednosti feature atributa i klasifikatornog atributa. Vrši se predikcija nad skupom podataka namenjneog za testiranje uz pozivanje metoda .predict(...) za svaki model klasifikacije ponaosob. Funkcija evaluate(...) daje tačnost po broju poklapanja instanci pre obrade predikcijom, i nakon nje.

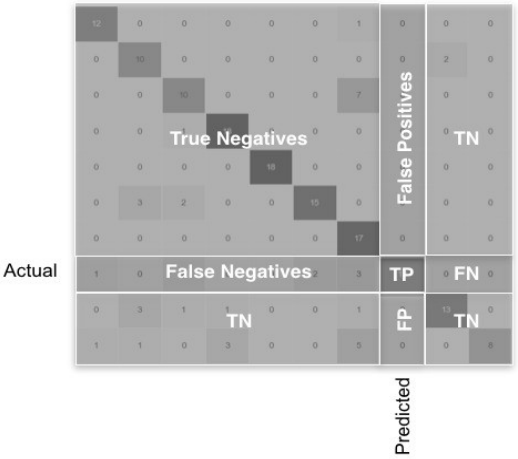
```
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import MultinomialNB
from sklearn.naive_bayes import BernoulliNB
from sklearn.naive_bayes import CategoricalNB

# ...
def evaluate(prediction, test):
        counter=0
        for i in range(len(prediction)):
         if prediction[i]==test[i]:
                    counter+=1
        accuracy = counter / len(test)
        return accuracy

GNB=GaussianNB()
GNB.fit(x_train,y_train)
prediction=GNB.predict(x_test)
accuracy=evaluate(prediction, list(y_test))
print(f'Gaussian Naive Bayes prediction accuracy : {accuracy}')

MNB=MultinomialNB()
MNB.fit(x_train,y_train)
prediction=MNB.predict(x_test)
accuracy=evaluate(prediction, list(y_test))
print(f'Multinomial Naive Bayes prediction accuracy : {accuracy}')

BNB=BernoulliNB()
BNB.fit(x_train,y_train)
prediction=BNB.predict(x_test)
accuracy=evaluate(prediction, list(y_test))
print(f'Bernoulli Naive Bayes prediction accuracy : {accuracy}')

CNB=CategoricalNB()
CNB.fit(x_train,y_train)
prediction=CNB.predict(x_test)
accuracy=evaluate(prediction, list(y_test))
print(f'Categorical Naive Bayes prediction accuracy : {accuracy}')
```

Implementacija 2: Implementacija primene raznovrsnih algoritama naivnog Bajesa

Obavljenim izvšavanjem ovog koda dobija se rezultat kao na slici 1. Gausov naivni Bajes daje najbolji rezultat od skoro 50% tačnosti, jer je baš namenjen za klasifikaciju nad kontinualnim vrednostima, dok multinomialni nad kategoričkim, Bernulijev nad binarnim feature vrednostima, kategorički namenjen nad kategoričkim feature vrednostima za koje odgovara kategorička distribucija verovatnoća[2]. Naivni Bajes algoritmi se zasnivaju nad računanja putem verovatnostne Bajesove formule $P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1,\ldots,x_n|y)}{P(x_1,\ldots,x_n)}$, $P_{aposterior} = \frac{P_{likelihood} * P_{aprior}}{P_{normalizationConstant}}$.

```
Gaussian Naive Bayes prediction accuracy :
0.4983164983164983
Multinomial Naive Bayes prediction accuracy :
0.3367003367003367
Bernoulli Naive Bayes prediction accuracy :
0.30303030303030304
Categorical Naive Bayes prediction accuracy :
0.30303030303030304
```

Slika 1: Tačnosti evaluirane za raznovrsne naivnog Bajesa algoritme



Slika 2: Osnova za nalaženje metrika nad konfuzionom matricom

Proračun metrika po svakoj vrednosti klasifikatornog atributa vrši se sagledanjem vrednosti koje su ključne. Senzitivitnosti (recall) ističe koliko validacijom model dobro predviđa u skladu 2 uparena skupa. Dok specifičnost ističe koliko se za 2 uparena skupa poklopljeno loše predviđa po modelu. Preciznost govori koliko je model sposoban da poklopi činjeničnu evaluaciju naspram nečinjenične sa obrascem skupa predvidjanog nad trening skupom. Tačnost u ovom slučaju govori koliko činjenični skup je pogodan naspram modela. F1-score računat po formuli $2 * \frac{precision * recall}{precision + recall}$, koji je kombinacija senzitivnosti i preciznosti. Prikazani su potrebni pojašnjeni postupci na slikama 2. i 3, a na slici 5 dat primer prikaza koji je zastupljen u ovom radu. Formiranje konfuzione matrice koja služi za prikaz (kao na slici 6.) zastupljenosti pri poklapanju skupa podataka nakon predikcija sa onim koji su dati pre predikcija, prikaz konfuzione matrice, zasebna izdvajanja metrika za svaku klasu zasebno se vrši u implementaciji 3 (slika 4.). Ovaj postupak je obavljen samo nakon obučavanja klasifikacijom Gausovim Naivnim Bajesom.

		True condition			
	Total population	Condition positive	Condition negative	Prevalence	Accuracy (ACC) $ACC = \frac{TP + TN}{TP + TN + FP + FN}$
Predicted condition	Predicted condition positive	True positive (TP), Power	False positive (FP), Type I error	Positive predictive value (PPV) (Precision) $PPV = \frac{TP}{TP + FP}$	False discovery rate (FDR)
	Predicted condition negative	False negative (FN), Type II error	True negative (TN)	False omission rate (FOR)	Negative predictive value (NPV)
		True positive rate (TPR) (Recall, Sensitivity, probability of detection) $TPR = \frac{TP}{TP + FN}$	False positive rate (FPR) (Fall-out, probability of false alarm) $FPR = \frac{FP}{FP + TN}$	Positive likelihood ratio (LR+)	Diagnostic odds ratio (DOR) / F1 score $F1 = \frac{2 * PPV * TPR}{PPV + TPR}$
		False negative rate (FNR) (Miss rate) $FNR = \frac{FN}{TP + FN}$	True negative rate (TNR) (Specificity (SPC), Selectivity) $TNR = \frac{TN}{FP + TN}$	Negative likelihood ratio (LR−)	

Slika 3: Način orjentisanja zarad nalaženja statističkih metrika

```
import seaborn
import matplotlib.pyplot as plt
import numpy as np

n_unique_instances_of_class = len(klasifikacije.data[klasifikacije.data.columns[-1]].unique())

def form_matrix(y_test, y_pred):
        matrix = np.zeros ((n_unique_instances_of_class, n_unique_instances_of_class))
        for i in range(len(y_test)):
                matrix[y_pred[i]][y_test[i]] += 1
        return matrix.astype('uint64')

def display_matrix(matrix):
        plt.figure()
        seaborn.heatmap(matrix, annot=True, cbar=False, cmap='Blues', fmt='d')
        plt.xlabel('true label')
        plt.ylabel('predicted label')
        plt.title('Confusion matrix')

def evaluate_metrics(matrix):
        temp = np.array(matrix)
        for i in range(n_unique_instances_of_class):
                TP = temp[i][i]
                FP = np.sum(temp[i]) - TP
                FN = np.sum(temp, axis=0)[i] - TP
                TN = np.sum(np.sum(temp)) - TP - FP - FN
        print(f'Evaluacija po Klasi {i} ima metrike:')
        print(f'TP {TP}, TN {TN}, FN {FN}, FP {FP}')

        TP = TP.item()
        FN = FN.item()
        FP = FP.item()
        TN = TN.item()

        if TP + FP != 0 and TP + FN != 0 and TN + FP != 0 and TN + FN != 0 and (TP != 0 or TN != 0):
                precision = TP / float(TP + FP)
                print('Precision: ' + str(round(precision, 2)))
                NPV = TN / float((TN + FN))
                print('Negative predictive value: ' + str(round(NPV, 2)))
                recall = TP / float((TP + FN))
                print('Recall :' + str(round(recall,2)))
                specificity = TN / float((TN + FP))
                print('Specificity :' + str(round(specificity,2)))
                if  0 != float((precision + recall)):
                        f1 = (2*precision * recall) / float((precision + recall))
                print('f1 score : ' + str(round(f1, 2)))
                print()
        else:
                print('metrics were unable to be calculated')
```

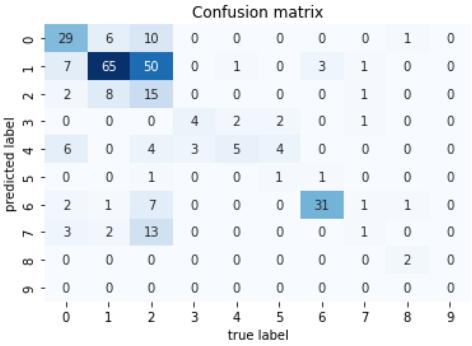
Implementacija 3: Formiranje konfuzione matrice, prikaz, zasebna izdvajanja metrika za svaku klasu zasebno

```
Evaluacija po Klasi 0 ima metrike:
TP 29, TN 231, FN 20, FP 17
Precision: 0.63
Negative predictive value: 0.92
Recall :0.59
Specificity :0.93
f1 score : 0.61

Evaluacija po Klasi 1 ima metrike:
TP 65, TN 153, FN 17, FP 62
Precision: 0.51
Negative predictive value: 0.9
Recall :0.79
Specificity :0.71
f1 score : 0.62

Evaluacija po Klasi 2 ima metrike:
TP 15, TN 186, FN 85, FP 11
Precision: 0.58
Negative predictive value: 0.69
Recall :0.15
Specificity :0.94
f1 score : 0.24

Evaluacija po Klasi 3 ima metrike:
TP 4, TN 285, FN 3, FP 5
Precision: 0.44
Negative predictive value: 0.99
Recall :0.57
Specificity :0.98
f1 score : 0.5

Evaluacija po Klasi 4 ima metrike:
TP 5, TN 272, FN 3, FP 17
Precision: 0.23
Negative predictive value: 0.99
Recall :0.62
Specificity :0.94
f1 score : 0.33

Evaluacija po Klasi 5 ima metrike:
TP 1, TN 288, FN 6, FP 2
Precision: 0.33
Negative predictive value: 0.98
Recall :0.14
Specificity :0.99
f1 score : 0.2

Evaluacija po Klasi 6 ima metrike:
TP 31, TN 250, FN 4, FP 12
Precision: 0.72
Negative predictive value: 0.98
Recall :0.89
Specificity :0.95
f1 score : 0.79

Evaluacija po Klasi 7 ima metrike:
TP 1, TN 274, FN 4, FP 18
Precision: 0.05
Negative predictive value: 0.99
Recall :0.2
Specificity :0.94
f1 score : 0.08

Evaluacija po Klasi 8 ima metrike:
TP 2, TN 293, FN 2, FP 0
Precision: 1.0
Negative predictive value: 0.99
Recall :0.5
Specificity :1.0
f1 score : 0.67

Evaluacija po Klasi 9 ima metrike:
TP 0, TN 297, FN 0, FP 0
metrics were unable to be calculated
```

Slika 4: Standardni izlaz za funkciju evaluate_metrics(...)

	precision	recall	f1-score	support
0	0.63	0.59	0.61	49
1	0.51	0.79	0.62	82
2	0.58	0.15	0.24	100
3	0.44	0.57	0.50	7
4	0.23	0.62	0.33	8
5	0.33	0.14	0.20	7
6	0.72	0.89	0.79	35
7	0.05	0.20	0.08	5
8	1.00	0.50	0.67	4
accuracy			0.52	297
macro avg	0.50	0.50	0.45	297
weighted avg	0.56	0.52	0.48	297

Slika 5: Standardni izlaz pri pozivanju classification_report(...)-a nad datim skupom podataka za obuku



Slika 6: Konfuziona matrica prikazana nakon predikcije nad skupom podataka za obučavanje

U implementaciji 4. dela funkcije evaluate_naive_bayes_custom_35_percent(...) se izvršava nova podela, ponovno kreiranje modela, evaluiranje i prikaz samo sa većom stopom količine instanci u skupu podataka za testiranje.

```
x_train, x_test, y_train, y_test = train_test_split(
        klasifikacije.data[klasifikacije.data.columns[:-1]],
        klasifikacije.data[klasifikacije.data.columns[-1]],
        test_size=0.35
        )
```

Implementacija 4: Povećanje stope količine instanci u skupu podataka za testiranje unutar evaluate_naive_bayes_custom_35_percent(...)

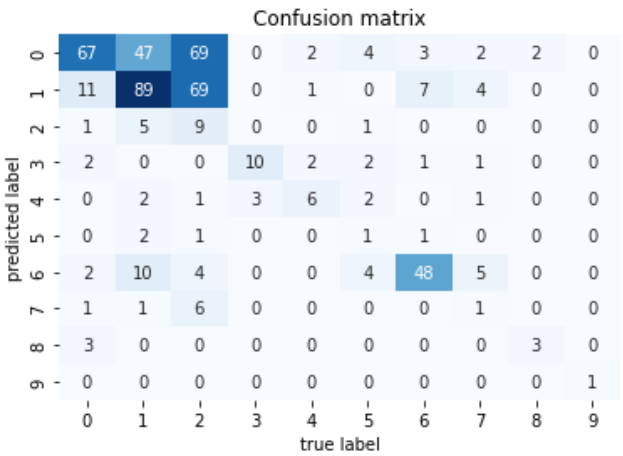
Dobija se identičan rezultat, gde se postupak nije proslavio mnogo bolje i u datim prikazima na slici 7. i 8. vidi se za zastupljenost i za ostale statističke metrike zasebnih vrednosti zasebnih instanci klasifikatornog atributa (na slikama su jednoznačne vrednosti enkodirane enumerisano) da rezultati nisu mnogo drugačije.

Na slici 5. i 7. u zadnja 3 reda se daju statističke vrednosti:

1. koliki je krajnji accuracy na 297 (na slici 5.) i 520 (na slici 7.) sagledanih instanci oba skupa (52%, a u drugom slučaju 45%) nakon primene klasifikacije Gausovog naivnog Bajesa,
2. macro avg - metrike (preciznost, senzitivnost, f1) se izračunavaju prosekom podjednakog udela svake vrednosti ponaosob.[3][4]
3. weighted avg - metrike se računaju tako što na prosek utiče težinski udeo po zastupljenosti vrednosti klasifikatornog atributa ponaosob.

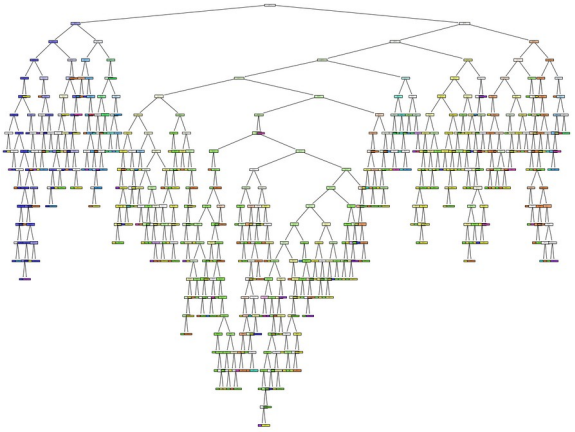
	precision	recall	f1-score	support
0	0.34	0.77	0.47	87
1	0.49	0.57	0.53	156
2	0.56	0.06	0.10	159
3	0.56	0.77	0.65	13
4	0.40	0.55	0.46	11
5	0.20	0.07	0.11	14
6	0.66	0.80	0.72	60
7	0.11	0.07	0.09	14
8	0.50	0.60	0.55	5
9	1.00	1.00	1.00	1
accuracy			0.45	520
macro avg	0.48	0.53	0.47	520
weighted avg	0.49	0.45	0.39	520

Slika 7: Prikaz nakon evaluate_naive_bayes_custom_35_percent(...) *za metrike i statističke vrednosti*



Slika 8: Prikaz konfuzione matrice nakon evaluate_naive_bayes_custom_35_percent(...)

Ovde u će se koristiti model klasifikacije (binarnog) stabla odlučivanja. Stablo se prožima od korena (neklasifikovanog skupa podataka) tako što se vrši obuhvatanje instanci u 2 zasebne klase za svaki nivo svakog podstabla, tu je kao objašnjenje data slika 9, kao i izlaz 2. Ovaj primer je dosta neoptimizovan, pa tu je i garancija da je došlo do procesa preučavanja (overfitting-a).



Slika 9: Decision Tree klasifikacija vizuelizovana za ovaj primer nakon obuke

```
1  from sklearn.tree import DecisionTreeClassifier
2  from sklearn.tree import export_text
3  from sklearn.tree import plot_tree
4
5  def show_text_representation(tree):
6      text_representation=export_text(tree)
7      print(text_representation)
8
9  def show_graphical_representation(tree):
10     fig = plt.figure(figsize=(25,20))
11     _ =plot_tree(tree, feature_names=klasifikacije.data.columns,
filled=True)
12
13 def basic_decision_tree(data):
14         ...
15         decision_tree_model = DecisionTreeClassifier()
16         decision_tree_model.fit(x_train, y_train)
17         predicted = decision_tree_model.predict(x_test)
18
19         # show_text_representation(decision_tree_model)
20         show_graphical_representation(decision_tree_model)
21         ...
```

Implementacija 5: Vrši se obuka, predikcija, prikaz klasifikacije stabla odlučivanja

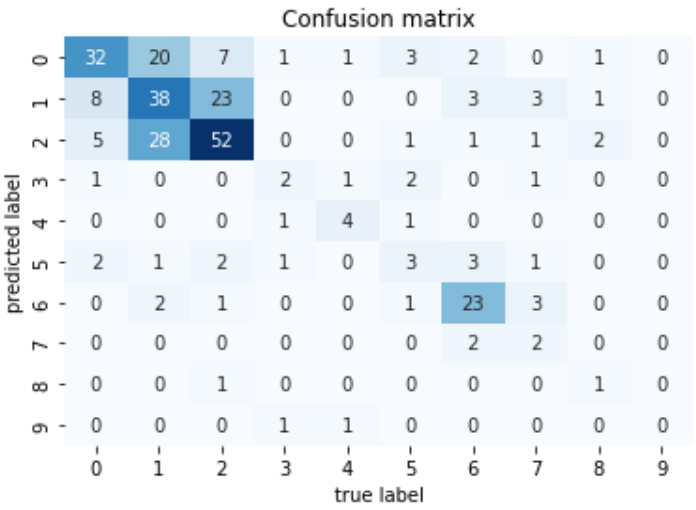
```
...
2  |  |  |--- feature_2 >  0.62
3  |  |  |  |--- feature_8 <= 0.23
4  |  |  |  |  |--- feature_1 <= 0.79
5  |  |  |  |  |  |--- feature_2 <= 0.69
6  |  |  |  |  |  |  |--- feature_2 <= 0.64
7  |  |  |  |  |  |  |  |--- class: 4
8  |  |  |  |  |  |  |--- feature_2 >  0.64
9  |  |  |  |  |  |  |  |--- feature_2 <= 0.66
10 |  |  |  |  |  |  |  |  |--- class: 0
11 |  |  |  |  |  |  |  |--- feature_2 >  0.66
12 |  |  |  |  |  |  |  |  |--- feature_1 <= 0.72
13 |  |  |  |  |  |  |  |  |  |--- class: 7
14 |  |  |  |  |  |  |  |  |--- feature_1 >  0.72
15 |  |  |  |  |  |  |  |  |  |--- class: 5
...
```

Izlaz 1: Tekstualni izdvojeni prikaz klasifikacije stablom odlučivanja, tj 1 podstabla

Obavljeno je izračunavanje metrika, kao i statističkih vrednosti prikazana na slici 10. Postignuta je tačnost od 52%.

	precision	recall	f1-score	support
0	0.48	0.67	0.56	48
1	0.50	0.43	0.46	89
2	0.58	0.60	0.59	86
3	0.29	0.33	0.31	6
4	0.67	0.57	0.62	7
5	0.23	0.27	0.25	11
6	0.77	0.68	0.72	34
7	0.50	0.18	0.27	11
8	0.50	0.20	0.29	5
9	0.00	0.00	0.00	0
accuracy			0.53	297
macro avg	0.45	0.39	0.41	297
weighted avg	0.54	0.53	0.53	297

Slika 11: Decision tree klasifikacija 0.2 test_size



Slika 10: Prikaz konfuzione matrice

Optimizacijom hiperparametara u implementaciji 6. ustupiće se približno izbegavanje overfitting-a, a i brže izvršavanje postupka. Ovde značajni parametri modela DecisionTreeClassifier sa kojima se vrše poređenja su:

- **max_depth** - dužina grana binarnog stabla od korena do lista kao usmerenog acikličnog grafa
- **min_samples_leaf** - je kriterijum za klasifikaciju da postane čvor list u stablu odlučivanja, podrazumevano je da bude 1.
- **min_samples_split** - kriterijum da se klasifikacija ostvari sa količinom instanci usvojenih, podrazumevano je da bude 2.[5]
- **class_weight** - konfiguriše sagledanja značajnosti klasifikatornih vrednosti instanci:
  - no_weights - za sve multiklasifikatorne vrednosti postavljena ista težina kao neka analogija značajnosti, a ona je u sastavu parametra class_weight kao liste.
  - weights - postavljeni na 0 da bi se obezbedila ravnomerna značajnost svi multiklasnih vrednosti, i ona je u sastavu parametra class_weight kao liste.

GridSearchCV je tehnika za objedinjavanje optimizacije hiperparametara i evaluiranja prosleđenog modela po svakoj kombinaciji konfigurisanoj. Cilj je naći najbolju moguću postavku hiperparametara. Definisan cross-validation se vrši kroz 5 stratifkovanih foldova. Fold-ove čine podeljeni skup podataka nad kojim se obučava model na 1/5 njegove veličine i formira validacioni skup podataka, gde se vrši višestruko obučavanje i ocenjivanja predikcija po validacionom skupu podataka nad umanjenim skupom podataka namenjenog za obuku - holdout estimation.[6] Obukom sa .fit(...), predikcijom nad vrednostima neklasifikatornih atributa zarad dobijanja vrednosti klasifikatornog atributa. Formirana je i prikazana konfuziona matrica, ispisane su metrike i statističke analize evaluacije obučenog modela naspram skupa podataka za testiranje. Dobijaju se parametri najbolje klasifikacije odabrane optimizacijom, a i dobija se sam model klasifikacije za buduću upotrebu.

```
from sklearn.model_selection import GridSearchCV
max_depth=[4,6,8,10]
min_samples_leaf=[2,5,7,10]
min_samples_split=[2,5,7,10]

num_classes=len(np.unique(klasifikacije.data[klasifikacije.data.columns[-1]]))

no_weights = dict()
for i in range(n_unique_instances_of_class):
        no_weights[i] = 0.1

weights = dict()
for i in range(n_unique_instances_of_class):
        weights[i] = 0
counts=[0 for i in range(n_unique_instances_of_class)]

values=list(klasifikacije.data[klasifikacije.data.columns[-1]])
for i in values:
        counts[i]+=1

for i in range(len(weights)):
    weights[i]=counts[i]/len(klasifikacije.data)
decision_tree_model = DecisionTreeClassifier()
class_weights=[no_weights , weights]

params={'max_depth':max_depth,
        'min_samples_leaf':min_samples_leaf,
        'min_samples_split':min_samples_split,
        'class_weight':class_weights}

model=GridSearchCV(decision_tree_model, params, cv=5)
model.fit(x_train,y_train)
predicted=model.predict(x_test)

params=model.best_params_
print()
print('best results were achieved with the following parameters: ')
print(params)
print()

dcc=model.best_estimator_
show_graphical_representation(dcc)

matrix = form_matrix(list(y_test), predicted)

display_matrix(matrix)

report = classification_report(list(y_test), predicted)
print(report)
```

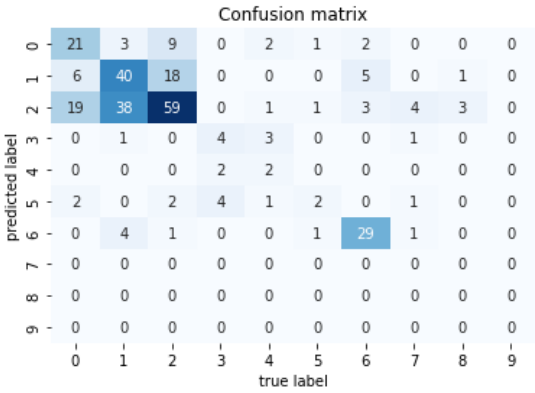
Implementacija 6: Vrši se optimizacija hiperparametara nad modelom klasifikacije stabla odlučivanja

Dobijeni su najbolje upotrebljeni parametri optimizacijom hiperparametara u odeljku za izlaz 2.

```
best results were achieved with the following parameters:
{'class_weight': {0: 0.1, 1: 0.1, 2: 0.1, 3: 0.1, 4: 0.1, 5: 0.1, 6: 0.1, 7: 0.1, 8: 0.1, 9: 0.1}, 'max_depth': 6,
'min_samples_leaf': 10, 'min_samples_split': 2}
```

Izlaz 2: Parametri nakon obrade optimizacije hiperparametara nad modelom klasifikacije stabla odlučivanja

Dati su i izlazi kao što je konfuziona matrica (slika 12.) i na standardnih izlaz statistička analiza sa metrikama (sa ne mnogo boljim rezultatima od ranijih) na slici 13. Vidi se kao i uvek donekle istaknuta zastupljenost ' CYT', ' NUC', ' ME3', ' MIT' vrednosti klasifikatornog atributa.

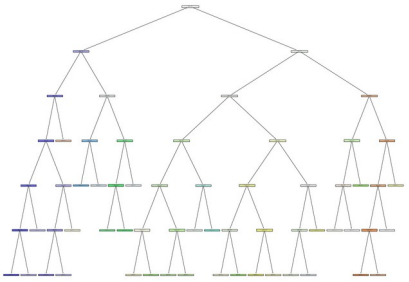


Slika 12: Konfuziona matrica najboljeg modela klasifikacije

	precision	recall	f1-score	support
0	0.55	0.44	0.49	48
1	0.57	0.47	0.51	86
2	0.46	0.66	0.54	89
3	0.44	0.40	0.42	10
4	0.50	0.22	0.31	9
5	0.17	0.40	0.24	5
6	0.81	0.74	0.77	39
7	0.00	0.00	0.00	7
8	0.00	0.00	0.00	4
accuracy			0.53	297
macro avg	0.39	0.37	0.36	297
weighted avg	0.53	0.53	0.52	297

Slika 13: Standadni izlaz najboljih parametara modelom klasifikacije stabla odlučivanja

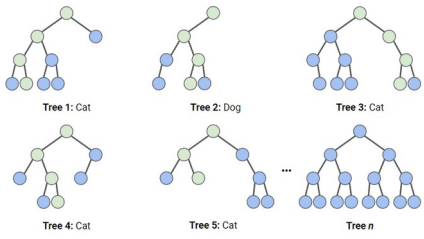
Iscrtano je modela klasifikacije stabla odlučivanja konfigurisanim sa najboljim parametrima na slici 14.



Slika 14: Rezultat stabla odlučivanja sa najboljim parametrima.

U implementaciji 7. će se koristiti RandomForestClassifier koji kao kod DecisionTreeClassifier kreira stablo odlučivanja koje ima takvu strukturu da su mu podstabla nasumično konstuisana (vršene klasifikacije nasumičnim redosledom. Pritom, verovatnoće zastupljenosti klasifikatornih vrednosti su tim faktorom nasumičnosti određene, kao što je na slici 15. prikazano. Biće obavljeno i optimizacijom hiperparametara:

- n_estimators - broj nasumičnih slučajeva potencijalnih stabala odlučivanja, naći najaktuelnije predvidjanje svih tih stabala odlučivanja, pa će to predviđanje biti usvojeno,
- max_depth - pominjano,
- min_samples_leaf - pominjano
- min_samples_split - pominjano
- class_weight - pominjano samo što je sada iskorišćena konfiguracija 'balanced' (težine vrednosti klasifikatornih atributa usaglašava proporcionalno prema učestalosti n_samples / (n_classes * np.bincount(y))), 'balanced_subsample' (koji je sličan 'balanced'-u samo na njega utiče po svakom podstablu pretumbavanje klasifikovanih instanci).



Slika 15: Kandidati stabla odlučivanja pri klasifikaciji RadnomForest do klasifikatornih vrednosti Cat/Dog[7]

U 3 stratifikovana fold-ova se obavlja cross-validation proces.

```
from sklearn.ensemble import RandomForestClassifier

n_estimators=[64, 128]
max_depth=[5,10,15]
min_samples_leaf=[2,4,6]
min_samples_split=[2,4,6]
class_weight=['balanced', 'balanced_subsample']

params={'n_estimators':n_estimators,
        'max_depth':max_depth,
        'min_samples_leaf':min_samples_leaf,
        'min_samples_split':min_samples_split,
        'class_weight':class_weight}

optimizer=GridSearchCV(RandomForestClassifier(), params, cv=3, verbose=True)
optimizer.fit(x_train,y_train)
predicted=optimizer.predict(x_test)

params=optimizer.best_params_
print('The best results were obtained using the following set of hyperparameters: ')
print(params)
print()

matrix = form_matrix(list(y_test), predicted)

display_matrix(matrix)

report = classification_report(list(y_test), predicted)
print(report)
```

Implementacija 7: Vrši se optimizacija hiperparametara, obuka nad najboljim parametrima, predikcija po skupu podataka za testiranje, prikaz metrika, konfuzione matrice

Nađeni najbolji parametri su istaknuti u odeljku izlaz 4.

```
The best results were obtained using the following set of hyperparameters:
{'class_weight': 'balanced', 'max_depth': 15, 'min_samples_leaf': 2, 'min_samples_split': 6, 'n_estimators': 64}
```

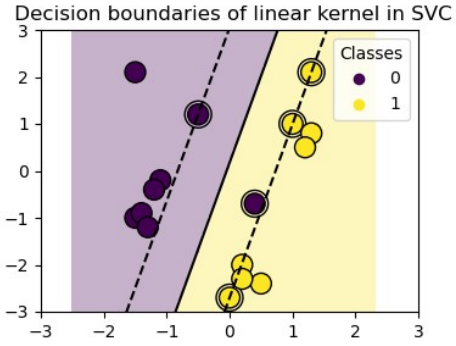
Izlaz 3: Nađeni najbolji parametri za ovaj slučaj pri klasifikaciji RadnomForest modelom

Ovo je do sada najbolja moguća izvedena klasifikacija **sa tačnošću od 70%**. Na slici 16. su dati rezultati metrika i statističkih analiza evaluacija najbolje parametrizovanog RandomForest modela.

	precision	recall	f1-score	support
0	0.77	0.65	0.71	55
1	0.70	0.71	0.70	85
2	0.69	0.68	0.68	97
3	0.83	0.83	0.83	6
4	0.43	0.60	0.50	5
5	0.54	0.58	0.56	12
6	0.74	0.93	0.82	27
7	0.50	0.40	0.44	5
8	1.00	0.60	0.75	5
9	0.00	0.00	0.00	0
accuracy			0.70	297
macro avg	0.62	0.60	0.60	297
weighted avg	0.70	0.70	0.70	297

Slika 16: classification_report(...) ispis za model RandomForest klasifikacije optimizovanih hiperparametara

U implementaciji 8. SVM (Support Vector Machine, mašina potpornih vektora) se koristi za klasifikaciju nad instancama skupa podataka za obuku. Pojedine instance će se istaknuti kao vektor. Cilj je naći najoptimalniju hiper-ravan koja će istaći 2 instance različitih vrednosti klasifikatornih atributa (label-a), a pritom su oni za svoje susetne instance, iste vrednosti klasikatornog atributa, diskriminantni (udaljeni), pa time uz marginu odrediti izopštenost klasa jednu od druge gledajući njihove instance predstavnike kao na slici 17 i te pomenute 2 instance će biti potporni vektori. Ciljani parametri ovog modela klasifikacije su:



Slika 17: prikaz klasifikazije linearnim SVM-om nad primerom binarne klasifikacije

- kernel - način na koji će se vršiti evaluacija i posmatranje features-a ravni (transformacija vršena u još veći dimenzionalni prostor) - kernel trik. Postoji pitanje da li su podaci linearno separabilni - ovaj model klasifikacije je pogodan za nelinearno separabilne probleme.[8] Ovaj skup podataka je linearno separabilan, pa ćemo koristiti samo linear i sigmoid postupke. Da koristimo postupke za nelinearno separabilne (rbf, poly) probleme, evaluacija bi uključila parcijalno diferenciranje i moglo bi doći do deljenja sa 0.[9]

$\langle x, x' \rangle$

- regularizacija - C parametrom, parametar kažnjavanja (pri netačnoj klasifikacij), ovime se ukazuje podnošljivost greške, jako utiče na kolebanje između ostvarivanja odluke o klasifikaciji ili kriterijuma odbačene klasifikacije. Utiče na veličinu margine koja se postavlja.

$\tanh(\gamma\langle x, x' \rangle + r)$

- gamma - koristan pri računanju rbf kernel postupka $\exp(-\gamma\|x - x'\|^2)$, polinomijalnog i sigmoid . Koristan za definisanje stope značajnosti instance iz skupa podataka obuke.

- decision_function_shape - kako će se izvršavati proračun ocene među vrednostima klasifikatornih atributa. Da li će ići 'sam protiv svih' ili 'jedan protiv drugog' ('ovo' ovaj 2. je pogodan za korišćenje prilkkom multiklasifikatorne klasifikacije).

```
from sklearn import svm

kernel=['linear', 'sigmoid']
decision_function_shape=['ovo']

params = {
        'decision_function_shape' : decision_function_shape,
        'kernel' : kernel
        }

#Create a svm Classifier
clf = GridSearchCV(svm.SVC(), params, cv=3, verbose=True)
clf.fit(x_train, y_train)
predicted = clf.predict(x_test)

params=clf.best_params_
print('The best results were obtained using the following set of hyperparameters: ')
print(params)
print()

matrix = form_matrix(list(y_test), predicted)

display_matrix(matrix)

report = classification_report(list(y_test), predicted)
print(report)
```

Implementacija 8: Vrši se optimizacija hiperparametara, obuka nad najboljim parametrima, predikcija po skupu podataka za testiranje, prikaz metrika, konfuzione matrice

Dobijeni parametari za najbolji model je prikazan na odeljku izlaza 4.

```
The best results were obtained using the following set of hyperparameters:
{'decision_function_shape': 'ovo', 'kernel': 'linear'}
```

Izlaz 4: Informacija o najboljim parametrima za model SVM-a

Dostignuta je tačnost od 59%. Ova klasifikacija se ipak preporučuje da se koristi umesto naivnog Bajesa, pošto je postupak evaluacije brži.

	precision	recall	f1-score	support
0	0.62	0.55	0.58	55
1	0.63	0.54	0.58	85
2	0.54	0.79	0.64	97
3	0.38	0.50	0.43	6
4	0.50	0.20	0.29	5
5	0.00	0.00	0.00	12
6	0.78	0.52	0.62	27
7	0.00	0.00	0.00	5
8	1.00	0.60	0.75	5
accuracy			0.59	297
macro avg	0.49	0.41	0.43	297
weighted avg	0.58	0.59	0.57	297

Slika 18: Postignuti rezultati istaknuti sa classification_report(...) za najbolji dobijeni model SVM-a

3. Zaključak

Kroz obavljene evaluacije modela klasifikacija Gausovog (GNB, pored poređenjenm tačnosti usputnih multinomijalnog, Bernulijevog, kategoričkog) Naivnog bajesa, korišćenja klasifikacija stabala odlučivanja, klasifikacija slučajnih šuma i klasifikacija mašina potpornih vektora, najbolji ishod je bio kod klasifikacija slučajnih šuma (RandomForestClassifier) sa najboljom tačnošću od 70%, 2. mašine potpornih vektora tačnosti od 59%, 3. stabla odlučivanja od 53%.

U okviru rada ne postoji zakljucak (posebna sekcija u kojoj su izdvojene informacije koji model se pokazao najbolje i na osnovu koje metrike, ne mora da ima puno sadrzaja).

Multiklasifikatorne vrednosti su bile enumerisane sa:

MIT 0	NUC 1	CYT 2	ME1 3	EXC 4	ME2 5	ME3 6	VAC 7	POX 8	ERL 9

Tabela 1: Presdtava svih vrednosti klasifikatornog atributa

Pokazani rezultati su obavljeni za sve obučene modele nad skupom podataka namenjenih za treinranje i podvrgnute testiranju predviđanjem skupom podataka namenjenih za testiranje.

GNB je imao najbolju preciznost za vrednost klasifikatornog atributa POX 1.00, recall za ME3 0.89 i za f1-score 0.79, support je bio najbolji za CYT 100.

GNB sa izmenjenom veličinom skupa podataka za testiranje sa 20% na 35% (isključivo za ovaj slučaj) imao je najbolju preciznost za vrednost klasifikatornog atributa POX 1.00, recall 1.00 i za f1-score 1 (ali je zato imao najmanji support od 1), support je bio najbolji za CYT 159 (što je najviši support do sada).

Klasifikacija stabla odlučivanja je doprinela da metrike budu najbolje po preciznosti za ME3 0.77, recall 0.68, f1 score 0.72, a support za NUC 89.

Klasifikacija stabla odlučivanja koje je optimizovano hiperparametrima je doprinela da metrike budu najbolje po preciznosti za ME3 0.81, recall 0.74, f1 score 0.77, a support za CYT 89.

Klasifikacija slučanih šuma imala je najbolju preciznost za ME1 0.83, recall za ME3 0.93, f1-score za ME1 0.83, support za CYT 97.

Klasifikacija mašine potpornih vektora imala je najbolju preciznost za POX 1.00, recall za CYT 0.79, f1-score za POX, najbolji support za CYT 97.

Literatura

1: UCI: Yeast dataset, https://archive.ics.uci.edu/dataset/110/yeast, Datum poslednjeg pristupa: 12.01.2024.
2: Naivni Bajes modeli klasifikacija, https://scikit-learn.org/stable/modules/naive_bayes.html, Datum poslednjeg pristupa: 12.01.2024.
3: Machine Learning Model Evaluation Metrics part 2: Multi-class classification, https://www.mariakhalusova.com/posts/2019-04-17-ml-model-evaluation-metrics-p2/, Datum poslednjeg pristupa: 12.01.2024.
4: Classification metrics: Multiclass and multilabel classification, https://scikit-learn.org/stable/modules/model_evaluation.html#multiclass-and-multilabel-classification, Datum poslednjeg pristupa: 12.01.2024.
5: DecisionTreeClassifier, https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html, Datum poslednjeg pristupa: 12.01.2024.
6: Cross-validation: evaluating estimator performance, https://scikit-learn.org/stable/modules/cross_validation.html#a-note-on-shuffling, Datum poslednjeg pristupa: 12.01.2024.
7: Random Forest Classification, https://www.datacamp.com/tutorial/random-forests-classifier-python, Datum poslednjeg pristupa: 12.01.2024.
8: SVM Tutorial, https://www.datacamp.com/tutorial/svm-classification-scikit-learn-python, Datum poslednjeg pristupa: 12.01.2024.
9: 1.4. Support Vector Machines, https://scikit-learn.org/stable/modules/svm.html#svm-classification, Datum poslednjeg pristupa: 12.01.2024.