Univerzitet u Kragujevcu

SEMINARSKI RAD

iz predmeta Neuronske mreže

Tema:

Klasifikacija napisanih slova korišćenjem neuronske mreze

Mentor : dr. prof. Vesna Ranković
Student : Željko Simić 3vi/2023

1. Postavka zadatka

Vrši se identifikacija svakog ogromnog broja crno-belih pravougaonih prikaza kroz piksele kao jednu od 26 velikih slova u engleskoj abecedi. Slike slova su zasnovane na 20 različitih fontova i svako slovo ponaosob nasumično izobličeno da bude u jednom među ustanovljenih 20 fontova i da proizvede datoteku od 20000 jedinstvenih primeraka (stimulacija, uputa, instanci)[1]. Svaki stimulus (instanca) konvertovan je u 16 atributa (statističkih momenata i broja incidentnih veza) koji su potom skalirani da se uklope u opseg celobrojnih vrednosti $[0,15]$.

Ulazni atributi:

1. **x-box** - horizontalna pozicija površine na slici
2. **y-box** - vertikalna pozicija površine na slici
3. **width** - širina površine na slici
4. **high** - visina površine na slici
5. **onpix** - broj zastupljenih uključenih piksela na površini slike kojim se izražava slovo
6. **x-bar** - središte uključenih piksela u povši nad x osom
7. **y-bar** - središte uključenih piksela u povši nad y osom
8. **x2-bar** - središte disperzije na x-osi
9. **y2-ba**r - središte disperzije na y-osi
10. **xybar** - središte korelacije xy
11. **x2ybr** - središte $x^2 * y$
12. **xy2br** - središte $x * y^2$
13. **x-ege** - središte incidentnih veza (susednih piksela) sleva na desno prisutnih zarad razaznavanja oblika slova
14. **x-egvy** - korelacija **x-ege** sa y
15. **y-ege** - središte incidentnih veza (susednih piksela) odozdo na gore
16. **yegvx** - korelacija **y-ege** sa x

Izlazni atribut:

1. **lettr** - klasifikatorni atribut sa vrednostima 'A', 'B', ..., 'Z'.

Implementacijom 1. izvršava se učitavanje ciljanog fajla, bez zaglavlja sa imenima atributa. Svaka linija je učitana pri privremeno rezervisanom pristupu toku za učitavanje datoteka na određenoj relativnoj putanji, sve do kompletnog učitavanja i konvertovanja svakog reda reda (primerka) u posebnu listu kolona iz niske karaktera po delimiteru ',', gde time je niska isparčana. Preskakanjem praznih redova.

```
from csv import reader
...
def load_csv(filename):
	dataset = list()
	with open(filename, 'r') as file:
		csv_reader = reader(file)
		for row in csv_reader:
			if not row:
				continue
			dataset.append(row)
	return dataset
...
filename = 'letter.csv'
dataset =  load_csv(filename)
```

Implementacija 1: Učitavanje i skladištenje sadržaja datoteke letter.csv

U implementaciji 2. po određenoj poziciji kolone vršiće se parsiranje obostrano trimovanog stringa od blanko karaktera u float.

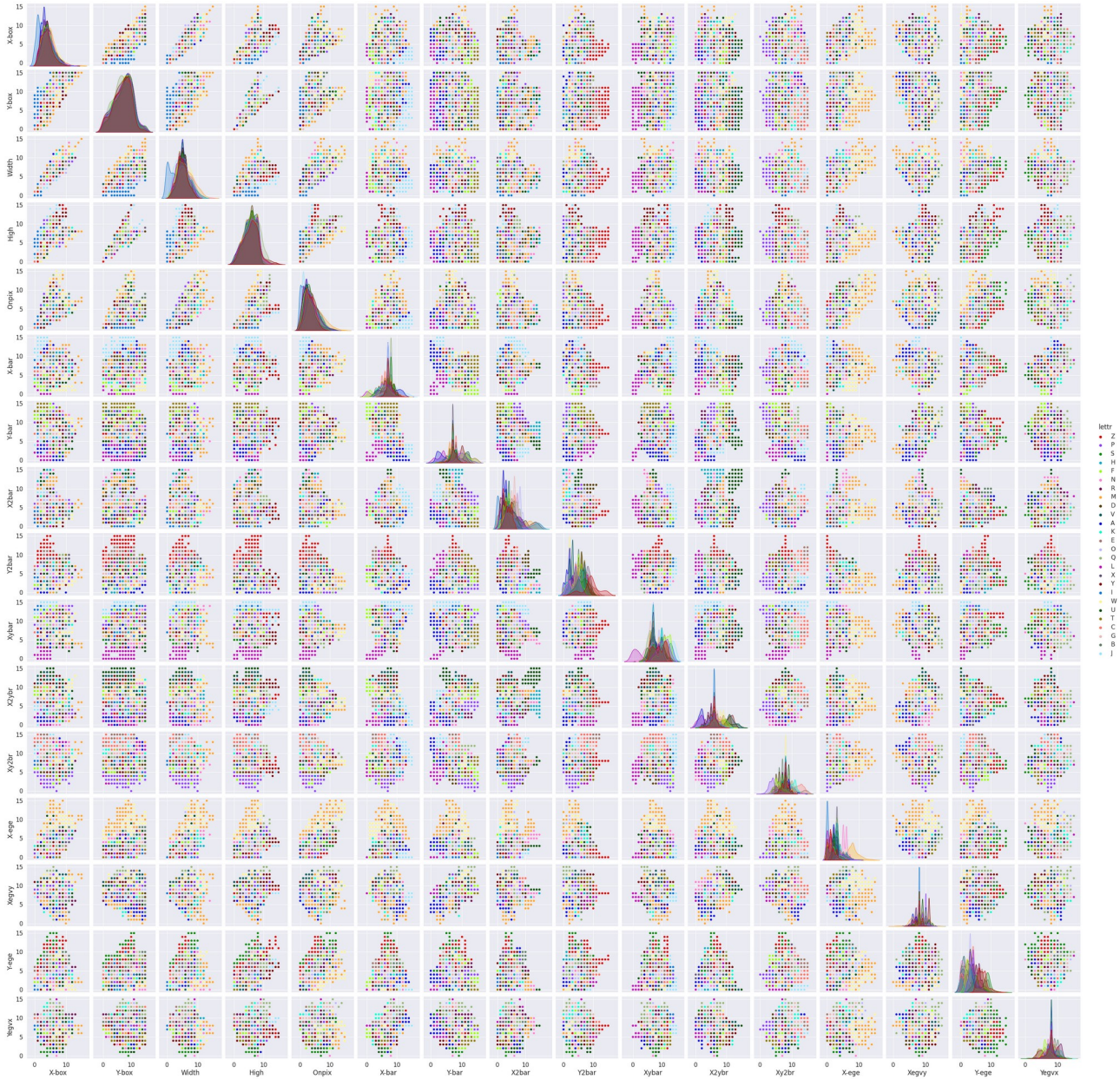
Sa druge strane, enkodiranje klasifikatornih atributa se vrši tako što se prikupe vrednosti primeraka u listu, a pritom pomoću objekta tipa torke (koji nema u sebi duplikate) ustupa jednoznačnost vrednosti za sagledanje i formira se asocijativni niz (rečnik) sa parovima ključ, vrednost. Ispuni da vrednost bude pozicija slova u pomenutoj torci, a ključ slovo - vrednost klasnog atributa. Tako se ažurira u skupu podataka kolona klasifikatornog atributa. A vrati asocijativni niz za svrhu dekodiranja.

```
def str_column_to_float(dataset, column):
	for row in dataset:
		row[column] = float(row[column].strip())

def str_column_to_int(dataset, column):
	class_values = [row[column] for row in dataset]
	unique = set(class_values)
	lookup = dict()
	for i, value in enumerate(unique):
		lookup[value] = i
	for row in dataset:
		row[column] = lookup[row[column]]
	return lookup
...
for i in range(len(dataset[0]) - 1):
	str_column_to_float(dataset, i)
...
str_column_to_int(dataset, len(dataset[0]) - 1)
```

Implementacija 2: Implementacija grupnog kastovanja u float neklasnih atributa, enkodiranja klasifikatornog atributa u int po vrednosti

2. Vizuelizacija

U implementaciji 3. vrši se vizuelizacija prikazana na slici 1. međusobne korelacije dva neklasifikatorna atributa naspram vrednosti klasifikatornog atributa.



Slika 1: Vizuelizacija međusobne korelacije dva neklasifikatorna atributa naspram vrednosti klasifikatornog atributa

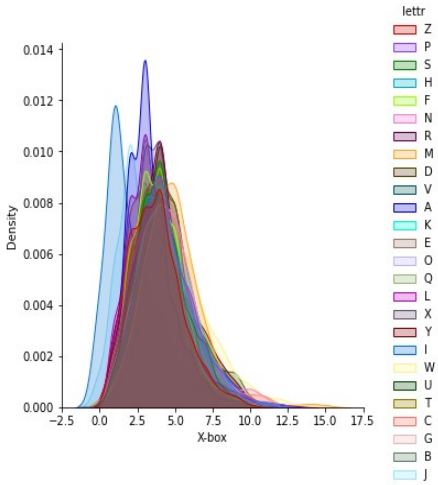
```
import numpy as np
import pandas as pd
import seaborn as sn
import matplotlib.pyplot as plt
import colorcet as cc
...
def visualize(dataset):
        df = pd.DataFrame(dataset,
                          columns=['X-box', 'Y-box', 'Width', 'High',
'Onpix', 'X-bar', 'Y-bar', 'X2bar', 'Y2bar', 'Xybar', 'X2ybr', 'Xy2br', 'X-ege',
'Xegvy', 'Y-ege', 'Yegvx', 'lettr'])
        palette = sns.color_palette(cc.glasbey, n_colors=26)
        sns.pairplot(df, hue="lettr", palette=palette);
```

Implementacija 3: Radi se prikaz prethodno dobijene ilustracije

Po dijagonali ovog prikaza se nalaze grafici funkciju verovatonoću zastupljenosti (raspodeljenosti, distirbucije, PDF - *probability distribution function*) primeraka sa vrednostima neklasifikatornih atributa svaki ponaosob se izražava kroz vrednosti klasifikatornog atributa. Na *slici 2 po implementaciji 4* ordinata pokazuje normalizovanu gustinu ($\frac{xbox - min(\forall xbox)}{max(\forall xbox) - min(\forall xbox)}$ - računat za svako xbox na apcisi ponaosob, pošto je implicitno podešen norm_hist=**True** [2] - što je korisno je pri komparaciji medju parametrima podataka pri tumačenju ogromnih skupova podataka). Za element slučajne promenljive u domenu [2.5, 5] i opsegu (0.008, 0.013) za većinu karaktera ustanovljena je X-box pozicija.

U obe prikazane implementacije funkcija se vrši konverzija iz objekta liste tipa u pandas.DataFrame tip vrednosti promenljive skupa podataka. Podešavanje palete boja zarad razaznavanja klasa vrednosti kojim se ističu slova.

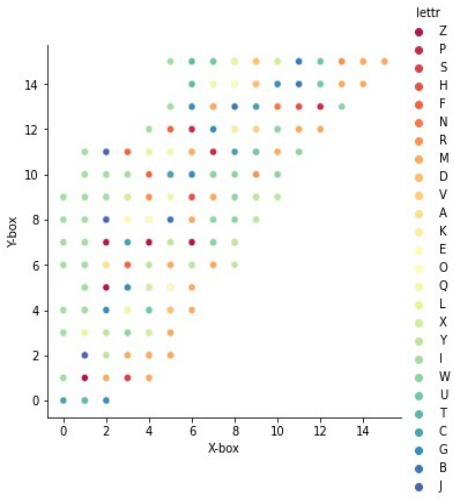
Van dijagonale mogu se uočiti posebno korelacije izmedju 2 neklasifikatornih atributa naspram klasifikatornog atributa zarad isticanja sličnosti slova po statističkoj karakteristici. Time je omogućeno sagledanje naspram čega je moguće obavljanje klasifikacija, pritom zaobilazeći problem preučavanja (overfitting-a da ne bi došlo do uvećanja greške ili umanjenja preciznosti), uočiti outlier instance, uočiti diskriminatorno udaljene klase naspram ostalih. Na primer, moguće je istaći da površina na slici na kojoj je napisano slovo po ilustraciji sa slike 3 - slovo 'I' uveliko je zastupljeno da mu površina na slici ima većinski zastupljen PDF za X-box (0, 2) na Y-box (4,6).



Slika 2: Ilustracija X-box atributa po zastupljenosti primeraka po vrednostima klasifikatornih atributa

```
def visualize_xbox_density(dataset):
        ...
        sns.displot(df,
                    hue="lettr",
                    palette=palette,
                    x=df['X-box'],
                    kind='kde',
                    fill=True)
```

Implementacija 4: Radi se prikaz ilustracija slike 2



Slika 3: Ilustracija korelacija X-box i Y-box atributa naspram lettr[3]

3. Pojašnjenje korišćene arhitekture i potrebe za njom

U implementaciji 5. vrši se konverzija skupa podataka u pandas.DataFrame sa ugradjivanjem oznaka svake od kolone po redosledu. Uzimaju se neklasni atributi kao kolone svih instanci ponaosob uz kastovanje u float i skladište se u promenljivu x. Skladišti se u promenljivu y klasifikatorni atribut kao kolona svake instance ponaosob. Vrši se enkodiranje liste, vrednosti y promenljive, pravi se dummy_Y promenljiva u koju se smešta vrednost binarne reprezentacije svake vrednosti iz y kao listu (npr. `2` kao rezultujuća vrednost enkodiranja binarizovana je u [`0`, `1`, `0`, ..., `0`]).

```
from sklearn.preprocessing import LabelEncoder
from keras.utils import np_utils
# ...
dataset = pd.DataFrame(dataset, columns=['X-box', 'Y-box', 'Width', 'High', 'Onpix', 'X-bar', 'Y-bar', 'X2bar', 'Y2bar', 'Xybar', 'X2ybr', 'Xy2br', 'X-ege', 'Xegvy', 'Y-ege', 'Yegvx', 'lettr'])

 y = dataset.iloc[:, -1]
# ...
encoder = LabelEncoder()
encoder.fit(y)
 encoded_Y = encoder.transform(y)
 dummy_Y = np_utils.to_categorical(encoded_Y)
```

Implementacija 5: Binarizacija i enkodiranje klasifikatornog atributa

U implementaciji 6. izvršava se nromalizacija podataka na neklasifikatornim atributima skupa podataka čime se dostiže skaliranje vrednosti tako što se preslika u opsegu [0,1] kao vrednost, time su preraspodeljene za lakšu upotrebu. Uzima se binarizovano enkodirano y u obzir. Radi se podela na skup za obučavanje i na skup za testiranje prema razmeri 80:20, a random_state naznačuje ponovno jednokratno pseudo-nasumično izvršavanje (sledećim pokretanjem isto će se izvršavati).

```
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler, LabelBinarizer
from sklearn.model_selection import train_test_split

   # ...
df_norm = dataset[dataset.columns[0:-1]].apply(
        lambda x : (x - x.min()) / (x.max() - x.min())
        )
 x = df_norm
 y = dummy_Y

x_train, x_test, y_train, y_test = train_test_split(
        x,y,
        test_size = 0.2,
        random_state = 101)
```

Implementacija 6: Normalizacija kolona kao neklasifikatornih atributa i vršenje podele skupa podataka za testiranje i obuku

U implementaciji 7. koristi se KerasClassifier pri klasifikaciji namenjen za pozivanje funkcije za kreiranje modela višeslojnog perceptrona neuronske mreže (pošto je multiklasni slučaj problema,

nalik XOR logičkom kolu, gde su vrednosti isključivo jednoznačne na izlazu, nisu linearno separabilne, tako se dostiže klasifikacija koja ima aktivacionu funkciju oblika višestepene nelinearne krive). Definiše se broj epoha kojim se uspostavlja broj prolaska kroz čitavu neuronsku mrežu, svakom epohom se vrši prikupljanje primeraka iz skupa podataka, potom ažuriraju težine ulaska u neuron i pragovi aktivacije (može dovesti do poboljšanih performansa svojom količinom pri optimizaciji, ali može doći i do overfitting-a). Definiše se veličina batch-a koji je jedno parčića skupa podataka za obuku uz rad određenog optimizatora - sa manjim njegovim brojem definiše se profinjenije koračanje kroz analizu i obuku, dok manja vrednost dovodi do nestabilne obuke (u ovom slučaju se koristi mini-batch mode veličina[4]). Tokom jedne epohe vrši se analiza svakog batch-a. Verbose daje na standardnom izlazu informaciju o vrednosti loss, accuracy, dosadašnjeg obavljenog obučavanja po epohi sa prikazom obrađenih batches-a.

```
from keras.wrappers.scikit_learn import KerasClassifier
 #...
model = KerasClassifier(build_fn = baseline_model,
                        epochs = 50,
                        batch_size = 256,
                        verbose = 1)
```

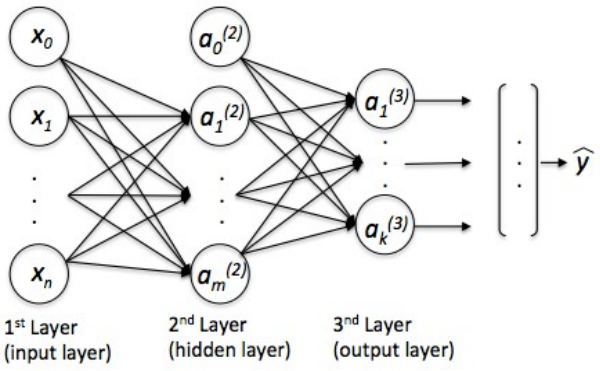
Implementacija 7: Instanciranje modela

Callback funkcija implementacije 8 koja je prosledjena KerasClassifier-u baseline_model dočekuje parametre:

1. optimizer (ako nije prosleđeno sam postavlja 'Adam' kao podrazumevano - prikaz postupka na slici 4.) - zaslužan za uspostavljanje bolje tačnosti i manjeg loss-a gde vrši ažuriranje težina, stope brzine obučavanja svakog od neurona svaki ima svoj poseban pristup. Npr. Adam se zasniva na metodologiji stohastičkog gradijentnog spusta (SGD, kao neka alternativa 'vanilla' feedforward i backpropagation metodu ažuriranja stanja neurona u perceptronu) u 2 momentuma konvergirajući iz jednog u drugi lokalni minimum (ne zarobljavajući se u njemu) sve do globalnog minimuma. Čak mini-batch SGD-em omogućeno je manja disperzija menju parametrima (izbegava prevelike oscilacije prilikom evaluiranja)[5].

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
$m_0 \leftarrow 0$ (Initialize 1st moment vector)
$v_0 \leftarrow 0$ (Initialize 2nd moment vector)
$t \leftarrow 0$ (Initialize timestep)
while θ_t not converged **do**
$t \leftarrow t + 1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
$\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
$\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$ (Update parameters)
end while
return θ_t (Resulting parameters)

Slika 4: Opis kako Adam funkcioniše



Slika 5: Predstavljanje 3-slojnog perceptrona

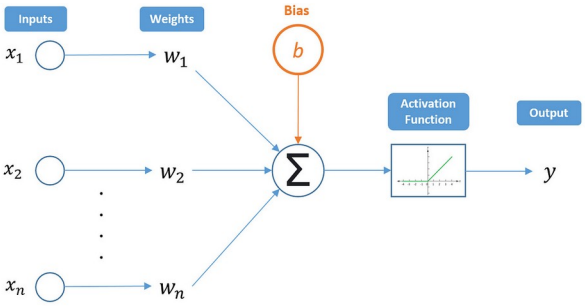
2. units (ako nije prosleđeno sam postavlja 10 kao podrazumevano) - zaslužno da ustanovi broj neurona u 1. (ulaznom) i u 2. (skrivenom, na ovom primeru jedinom) sloju 3-slojnog perceptrona.

3. init (ako nije prosleđeno sam postavlja 'normal' kao podrazumevano) - zaslužan za uspostavljanje početnih stanja težine ulaza neurona, kao i željenih vrednosti. Parametar ove funkcije za vrednost normal, tj. RandomNormal daje stanje neurona za ceo višeslojni perceptron sa središnjom vrednošću 0.0, i standardnom devijacijom 0.05 kao podrazumevanom.[6]

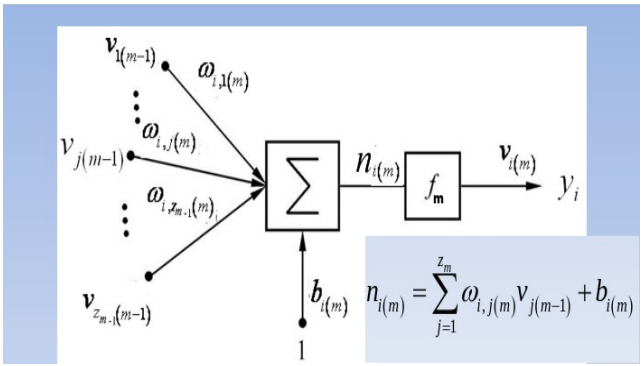
4. activation (ako nije prosleđeno sam postavlja 'relu' kao podrazumevano) - zaslužan parametar za uspostavljanje aktivacione funkcije za neurone prva dva sloja 3-slojnog perceptrona neuronske mreže. Njen oblik uz kombinaciju ostalih aktivacionih funkcija drugih neurona može da se odazire na kvalitet performansi evaluacije izlaznih vrednosti. Na ovom primeru gde je **ReLU(n) =** $\begin{cases} 0, n \leq 0 \\ n, n > 0 \end{cases}$, $n = (\sum_{i=1}^{n} w_i x_i) - b$, dat kao podrazumevana aktivaciona funkcija

   koja ustupa mogućnost da se obezbedi opstanak gradijenta pri evaluaciji (pogodan za razređenost stanja težina, ciljanih vrednosti i pragova aktivacija).[7]

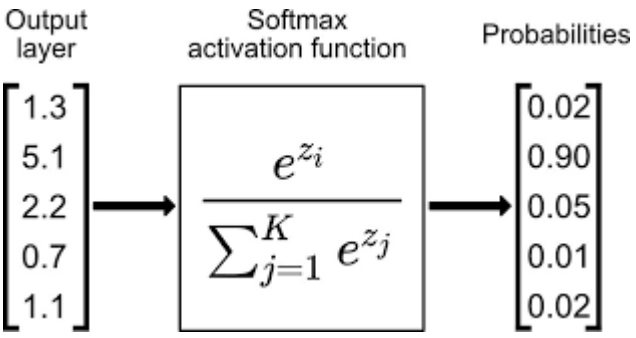
   Ilustracija data na slici 7. za poseban primer jedan neuron jednoslojnog perceptrona neuronske mreže zarad razjašnjenja, a zarad približenja tematici dat je opšti postupak i prikaz jednog neurona u višeslojnom perceptronu neuronske mreže.



Slika 7: Dat 1-slojni perceptron sa 1 neuronom u sebi kao opšti primer rada ReLU funkcije po modelu McCulloch i Pitts-a



Slika 6: Opšti slučaj prikaza modela statičkog neurona na poziciji i za sloj m sa formulom parametra za aktivacionu funkciju



Slika 8: Vizuelno demonstriranje softmax aktivacione funkcije

Vrši se instanciranje objekta Sequential klase kojom se dostiže omogućeno podešavanje dostupnih tenzora za ulaze i izlaze svakog uspostavljenog sloja koji se nagomilava u njemu, pa tako omogući realizacija višeslojnog perceptrona. Dodaju se inkrementalno slojevi skupa neurona (.add(...)) u pomenuti ovjekat, a svaki sloj se konstruiše uz pomoć konstruktora klase Dense čiji su parametri: broj neurona koje sadrži u sebi, broj ulaza u neurone (samo za ulazni sloj - broj predstavlja količinu kolona kao neklasifikatornih atributa), način početnog nameštanja stanja (težina veza ulaza, aktivacionih pragova, ciljanih vrednosti) i aktivacione funkcije za sve neurone sloja koji daju zaseban izlaz iz neurona. Na zadnjem sloju je postavljen broj neurona (gde je zaslužna ona binarizovana enkodirana kolona koja prezentuje klasifikatorni atribut) i 'softmax' aktivaciona funkcija (prikazana ideja postupka na slici 8.) kao pogodnu za multinomijalnu verovatnosnu raspodelu (pogodnu za kategoričke višebrojne vrednosti klasifikatornog atributa, a daje na izlaz verovatnoću eksponovanih vrednosti vektora parametra unutar njega od aktivacione funkcije kao novi vektor što je i izlaz vektora)[8].

Dropout se koristi prilikom obuke da bi se njime zaobišla problematika overfittinga (preučavanja zarad ubrajanja udaljenijih instanci po svojim neklasifikatornim vrednostima atributa naspram ubrojanih), vršenjem anuliranja proizvoljnog ulaza neurona po nekoj stopi (*rate*), na ovom primeru 0.2), a oni preostali ulazi neurona dobijaju skaliranje po formuli $\frac{1}{1-rate}$ tako da suma ulaza je uveliko neizmenjena. Na kraju se vrši konfiguracija modela za vršenje funkcije analize gubitka, funkcije optimizacije, isticanja metrika analize - tačnost. Funkcija analize gubitka može doprineti tačnosti prilikom ažuriranja stanja neurona prema nekim ciljanim vrednostima. Loss metoda 'categorical_crossentropy' daje meru analizom odnosa izmedju zastupljenosti tačnih vrednosti i

prediktovanih vrednosti po njihovim neusaglašavanjima nakon evaluacija po svakoj vrednosti klasifikatornog atributa.[9]

$$-\frac{1}{brojIzlaza} * \sum_{i=1}^{brojIzlaza} ytrue_i * log(ypred_i) + (1 - ytrue_i) * log(1 - ypred_i)$$

Metrika tačnosti daje odnos poklopnjenih (predvidjenih i tačnih) evaluacija naspram sveukupnog obavljenog evaluiranja po zastupljenosti vrednosti klasifikatornih atributa.

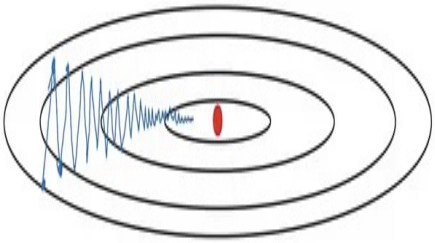
```
from keras.models import Sequential
from keras.layers import Dense, Dropout
# ...
def baseline_model(optimizer='Adam', units=10, init='normal', activation='relu'):
    model = Sequential()
    model.add(Dense(units, input_dim = 16, kernel_initializer=init, activation = activation))
    model.add(Dropout(0.2))
    model.add(Dense(units, kernel_initializer=init, activation = activation))
    model.add(Dropout(0.2))
    model.add(Dense(26, kernel_initializer=init, activation = 'softmax'))
    model.add(Dropout(0.2))
    model.compile(loss = 'categorical_crossentropy',
                  optimizer = optimizer,
                  metrics = ['accuracy'])
    return model
```

Implementacija 8: Građenje višeslojnog perceptrona i podešavanje konfiguracije

GridSearchCV je tehnika za objedinjavanje optimizacije hiperparametara i evaluiranja prosleđenog modela po svakoj kombinaciji konfigurisanoj. Prilagođavanje hiperparametara se omogućava unapređenje generalizacije i performansi perceptrona (preko stope brzine obuke, veličine batch-a, odabira aktivacionih funkcija, itd.) - zaobilaženje overfitting-a, otkriće neotkrivenih podataka, postignuta pojačana tačnost. Cilj je naći najbolju moguću postavku hiperparametara. Definisan cross-validation KFold parametar omogućava da nad skupom podataka namenjenim za obuku se izvrši pojačanje performansi modela tako što se prolazi kroz dodatan proces generacijskih ponovnih podela (fold njih k generacija podela, na n_splits podelom batches-a) uz pretumbavanje podataka pre nego što se generiše za jednu generaciju podele 1 batch ide za validacioni skup i k-1 batches-a ide skupu podataka namenjenim za obuku.

Optimizeri koji su ovde zastupljeni kao deo hiperparametra su:

1. SGD - obični stohastički gradijentni spust (stochastic gradient descent)
2. RMSprop - sličan algoritmu *standardnog gradijentnog spusta sa momentumom* (na slici 10. prikazan onaj bez momentuma). Otpisuje oscilacije u vertikalnom pravcu. Stoga je moguće povisiti stopu brzine obuke da bi se algoritam kretao većim koracima prema kovergenciji u horizontalnom pravcu. Drugačije se računa gradijent naspram algoritma gradijentnog spusta. [10] Ideja postupka prikazana na slici 9.



Slika 10: Standardni gradijentni spust

$$v_{dw} = \beta \cdot v_{dw} + (1-\beta) \cdot dw^2$$

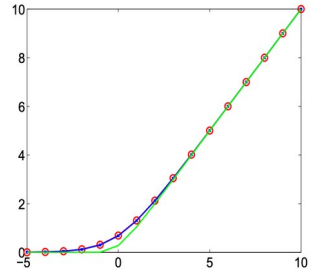
$$v_{db} = \beta \cdot v_{dw} + (1-\beta) \cdot db^2$$

$$W = W - \alpha \cdot \frac{dw}{\sqrt{v_{dw}} + \epsilon}$$

$$b = b - \alpha \cdot \frac{db}{\sqrt{v_{db}} + \epsilon}$$

Slika 9: formule za RMSprop

3. Adagrad - SGD bez koncepta momentuma momentuma sa idejom da ima različite stope brzine obuke prilikom svake iteracije sa razlogom da razređenost parametara kao neklasifikatornih atributa je retko učestala i sa tim njoj je potrebna visoka stopa obuke, naspram gušćih parametara kao neklasifikatornih atributa. Nije zgodno ručno menjati stopu brzine obuke, jer nekom modelu može obuka da bude redudantna i stopa obuke dosegne vrednost 0.[11]
4. Adadelta - proširenje algoritma Adagrad, gde se ograničava eksponencijalno uvećanje koeficijenta kojim se menja stopa brzine obuke.
5. Adam - pominjan SGD sa 2 momentuma (SGD sa momentumom, Adadelta usklađujuća stopa brzine obuke)
6. Adamax - koristi L-infinity normu gradijenta (vrlo zgodno pri ogromnoj razređenosti gradijenata). Odsutnost 2. momentuma uručuje bržu konvergenciju i manje memorijske zahtevnosti.[12]
7. NAdam - verzija Adam optimizacije koji koristi Nesterov trik, gde se dobija brža konvergencija.[13]



Slika 11: Softplus grafik

Aktivacione funkcije koje su ovde zastupljene kao deo hiperparametra su:

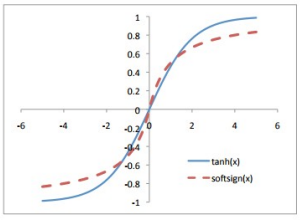
1. softmax - pomenuti
2. softplus - posmatran kao glatkija verzija ReLU, po slici 11.

$$f(x) = \log(1 + \exp(x))$$

3. softsign - više zahtevniji od tanh ili simgoida. Prikazan je njegov grafik na slici 12.

$$softsign(x) = \frac{x}{1 + |x|}$$

4. ReLU - pomenuti



Slika 12: Grafik softsign funkcije naspram tangensa hiperboličkog

5. tanh - opseg mu je između (-1, 1), a centar na 0, isprekidan grafik je prikazan na slici 12. Mana je obezbeđivanje opstajanja gradijenata.[14]
6. sigmoid - zgodan pri binarnim klasifikacijama, zgodan za evaluacije nelinearne prirode. [15]
$$\frac{1}{1+e^{-x}}$$
7. hard_sigmoid - brži za račun od običnog sigmoida, nije diferencijabilna.

$$f(x) = \max\left(0, \min\left(1, \frac{(x+1)}{2}\right)\right)$$

8. linear - identitet ulaza ide na izlaz - nije korišćena na skrivenim slojevima i izlazom sloju. [16]

Sa .fit(...) se uspostavlja evaluacija nad podacima, takođe se na ovom primeru stavlja test skup podataka kao skup podataka namenjen za validaciju. Nakon svih evaluacija dopremaju se nazivi podataka najboljeg hiperparametra i njegov score koji je na opsegu od (0, 1]. Prikazana je implementacija u odeljku za implementaciju 9.

```
from sklearn.model_selection import Kfold, GridSearchCV
#...
kfold = KFold(n_splits = 3,
                    shuffle = True)
optimizers = ['SGD', 'RMSprop', 'Adagrad', 'Adadelta', 'Adam', 'Adamax', 'Nadam']
activations = ['softmax', 'softplus', 'softsign', 'relu', 'tanh', 'sigmoid', 'hard_sigmoid', 'linear']
init = ['normal']
param_grid = {
    'units': [5, 10, 15],
     'init': init,
     'activation': activations,
     'optimizer' : optimizers
 }
grid = GridSearchCV(estimator=model,
                                param_grid=param_grid,
                                cv= kfold)
grid_result = grid.fit(x_train, y_train,
                            validation_data = (x_test, y_test))
print("Best Parameters: ", grid_result.best_params_)
print("Best Score: ", grid_result.best_score_)
```

Implementacija 9: Konfiguracija optimizacija hiperparametara za bolju generalizaciju i performanse nad skupom podataka namenjenom za obuku

Moguće je ispratiti po implementaciji 10. kroz svaku epohu stanje tačnosti i gubitka uspostavljenog obučavanja i njegovog odaziranja pri radu sa validacionim skupom podataka.

```
import matplotlib.pyplot as plt

#...
history = grid.best_estimator_.model.history

plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Accuracy')
plt.legend(['Train', 'validation'])
plt.show()

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Loss')
plt.legend(['Train', 'validation'])
plt.show()
```

Implementacija 10: Vrši se prikaz 2 grafika u 2 plota naspram najboljeg hiperparametra

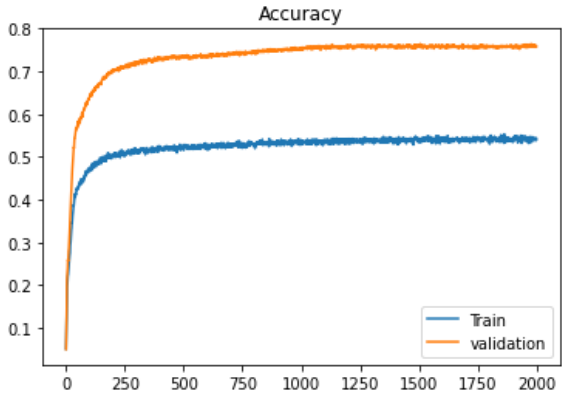
Vrši se predikcija nad skupom za obučavanje po implementaciji 11. Po toj predikciji se generiše konfuziona matrica kojom se sagledaju odnos stvarnih vrednosti naspram predviđenih vrednosti po klasifikatornom atributu. Pored konfuzione matrice generišu se metrike preciznost, senzitivnost, f1-score, support (zastupljenost), prosečnost.

```
import seaborn as sns
from sklearn.metrics import confusion_matrix, classification_report
# ...
predict_train = grid_result.predict(x_train)
print(classification_report(y_train.argmax(axis=1),
                           predict_train))

cm_train = confusion_matrix(y_train.argmax(axis=1),
                            predict_train)
fig, ax = plt.subplots(figsize=(26,26))
ax = sns.heatmap(cm_train, annot = True, cmap = 'Blues', cbar=False, linewidths=.5, ax=ax)
ax.set_xlabel('Predicted Values')
ax.set_ylabel('Actual Values')
plt.show()
# ima i za test skup pravljen classification_report i konfuziona matrica...
```

Implementacija 11: Vrši se prikaz statističkih metrika, kao i konfuzione matrice

4. Statističke metrike i doneti zaključci

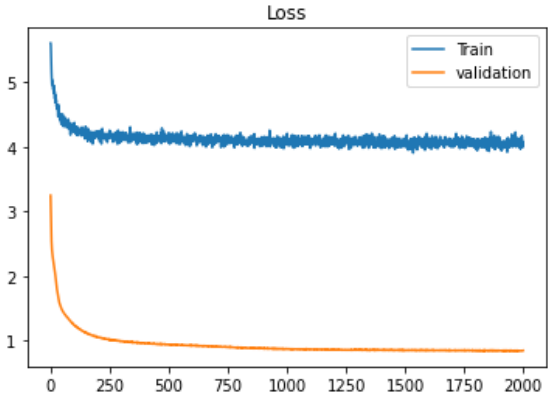
Za definisane parametre **2000 epoha** u **2 stratifikovana folda** cross-validation procesa, optimizacijom **NAdam**, aktivacionom funkcijom **tanh**, sa **15 neurona** ulaznog sloja i skrivenog sloja, pozivom funkcije evaluate_statistic_metrics_custom(x,y)**.** Dobija se ovakav proračun metrika modela 3-slojnog perceptrona neuronske mreže. Gde tačnost je vrednovana po statističkim vrednostima metrika poklapanja predvidjanja po proveri obuke modela naspram skupa koji se validira, tj. u ovom slučaju test skup.



Slika 13: Tačnost sračunata kroz

GridSearchCV je accuracy sračunao po formuli $\texttt{accuracy}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} 1(\hat{y}_i = y_i)$[17], gde je l karateristična funkcija i sa prikazom na slici 13.

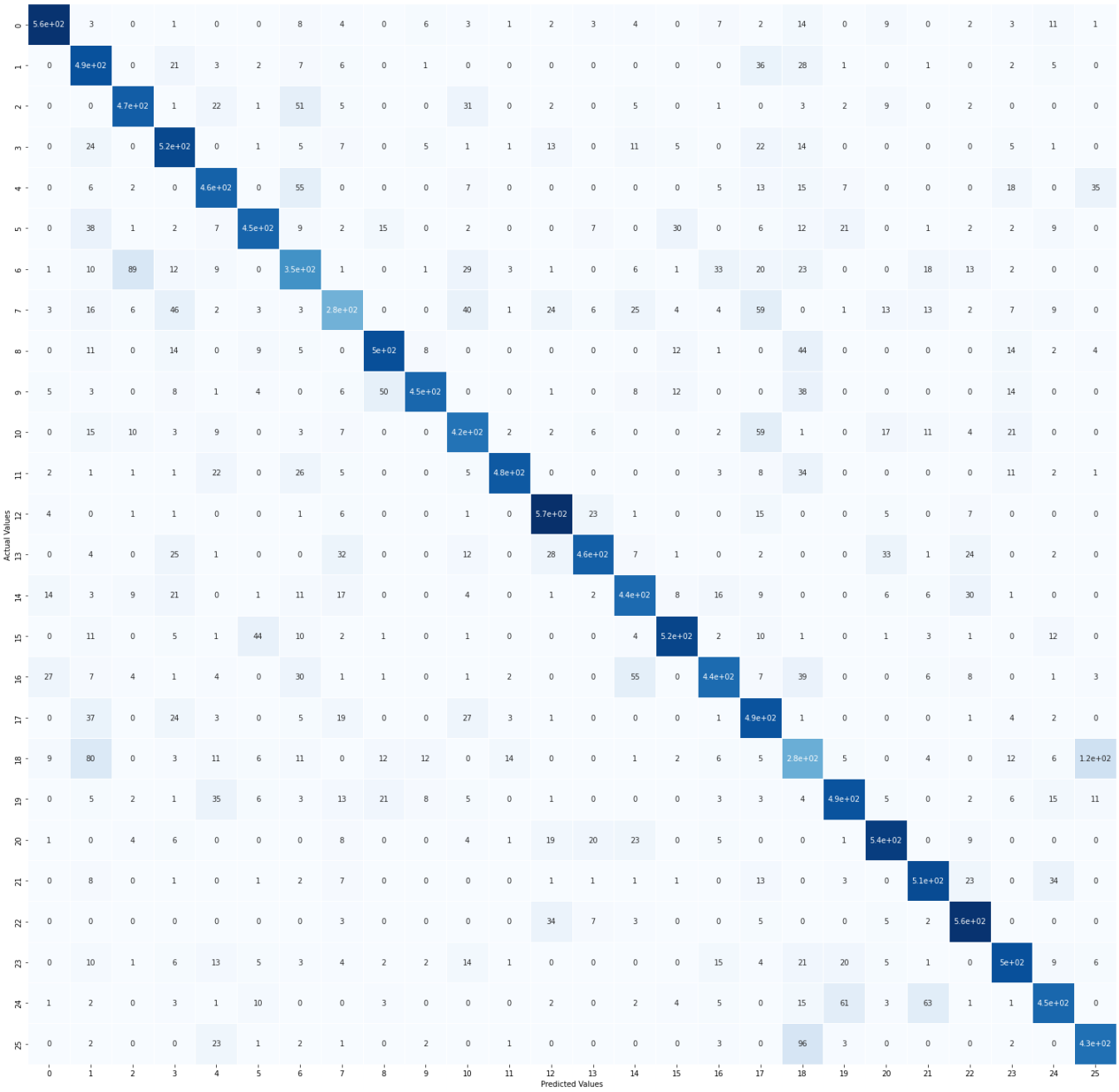
Loss koji je prikazan na slici 14 računa se po formuli $L_{\text{Hinge}}(y, w) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} \max\{1 + \hat{w}_{i,y_i} - w_{i,y_i}, 0\}$[18].
Ove formule se evaluiraju kroz svaku epohu.



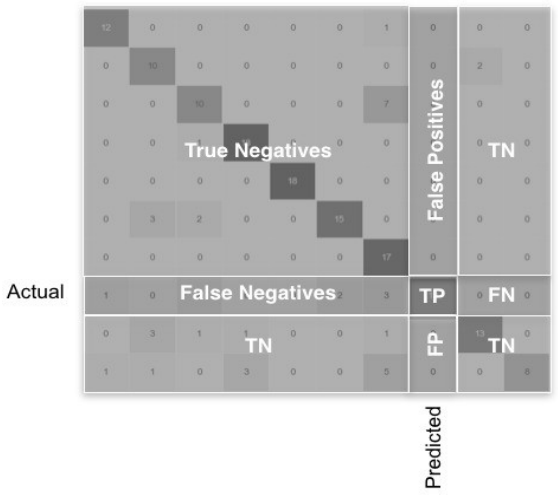
Slika 14: Gubitak sračunat kroz GridSearchCV za dati primer

Može se uočiti da metodom probe i greške da test skup se poklapa sa obukama modela više nego sam trening skup! Moj zaključak je da se samo zbog toga što je test skup manji, a i sličan lagodnom iskustvu modela. Dok trening skup je glomazniji, ima svoje diskriminatorne vrednost u odnosu na druge, pa i sam model kad se nad njime obučava daje jaču prioretizaciju na obrasce primeraka u odnosu na druge. Za loss funkciju takođe je čak apsurdniji slučaj gde proračun paralelno višestruko udaljen za proračun nad trening skupom, nego nad test skupom.

Kroz metričko sagledanje gleda se 'činjenični' skup, naspram skupa predviđanja nad skupom za obuku ulaznih atributa instanci (uz GridSearchCV.predict(...)), a naspram njih se formira matrica zastupljenosti vrednosti klasifikatornog atributa u odnosu oba skupa - konfuziona matrica, prikaz postavljen na slici 15. Treba uočiti da su vrednosti klasifikatornih atributa enumerisane enkodiranjem.

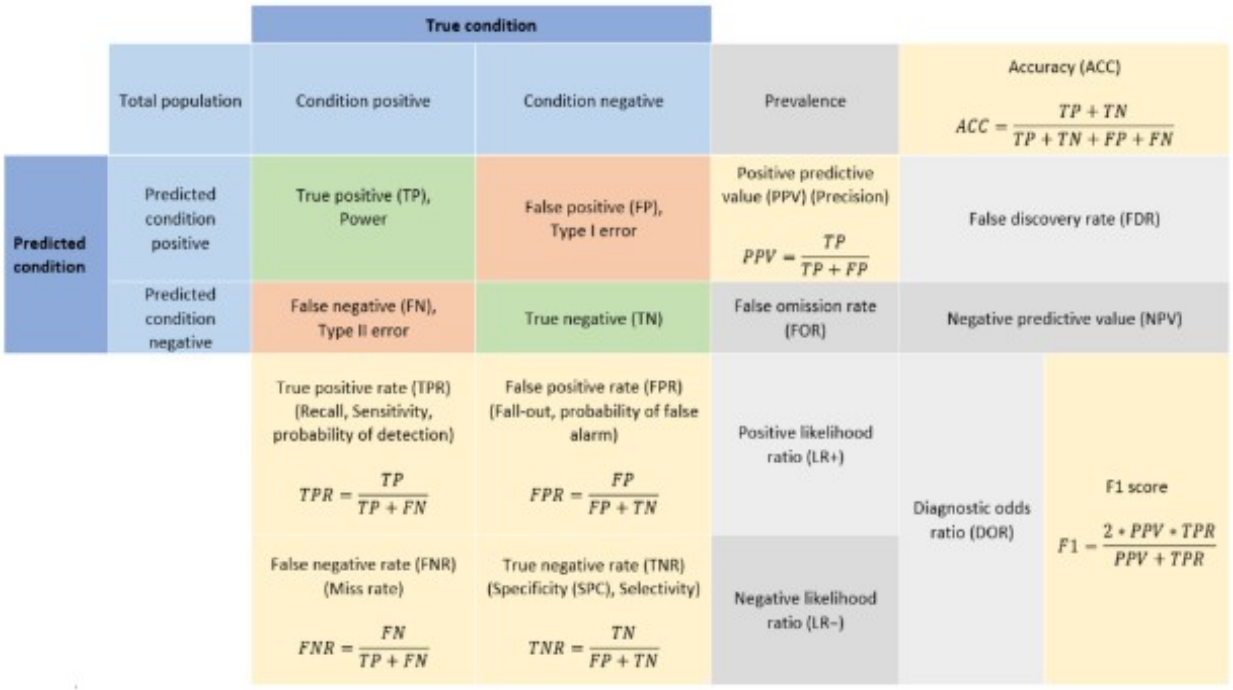


Slika 15: Konfuziona matrica za dati primer generisana seaborn heatmapom



Slika 16: 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 16. i 17.



Slika 17: Način orjentisanja zarad nalaženja statističkih metrika

Ovde na slici 18. izvodi se proračun nad predviđanjem skupom za obučavanje modela za svaku vrednost klasifikatornog atributa (feature-a), implementiranog po implementaciji 12. Može se uočiti da za slova L(11.), J(9.), A(0.) preciznost je jako visoka, za X(22.), M(12.), A(0.) je senzitivnost jako visoka, a sve na kraju A(0.), L(11.), M(12.), X(22.) imaju jak f1-score.

Pri predikciji sva slova su bila ravnomerno količinski zastupljena (koja se nalaze na dijagonali konfuzione matrice).

U zadnja 3 reda se daje:

1. koliki je krajnji accuracy na 16000 sagledanih instanci oba skupa (76%),
2. macro avg - metrike (preciznost, senzitivnost, f1) se izračunavaju prosekom podjednakog udela svake vrednosti ponaosob.[19][20]
3. weighted avg - metrike se računaju tako što na prosek utiče težinski udeo po zastupljenosti vrednosti klasifikatornog atributa ponaosob.

```
val_accuracy: 0.7582
Best Parameters:  {'activation': 'tanh', 'init':
'normal', 'optimizer': 'Nadam', 'units': 15}
Best Score:  0.750187486410141
500/500 [==============================] - 1s 2ms/step
              precision    recall  f1-score   support

           0       0.89      0.87      0.88       643
           1       0.63      0.81      0.71       607
           2       0.78      0.78      0.78       603
           3       0.71      0.82      0.76       630
           4       0.73      0.74      0.74       621
           5       0.83      0.73      0.78       620
           6       0.59      0.57      0.58       626
           7       0.64      0.49      0.55       563
           8       0.83      0.80      0.81       623
           9       0.91      0.75      0.82       596
          10       0.69      0.71      0.70       592
          11       0.94      0.80      0.86       600
          12       0.81      0.90      0.85       632
          13       0.86      0.73      0.79       636
          14       0.74      0.73      0.74       599
          15       0.87      0.83      0.85       630
          16       0.80      0.69      0.74       634
          17       0.62      0.79      0.70       619
          18       0.41      0.47      0.44       600
          19       0.80      0.77      0.78       639
          20       0.83      0.84      0.83       637
          21       0.80      0.84      0.82       604
          22       0.81      0.90      0.85       619
          23       0.80      0.78      0.79       640
          24       0.79      0.72      0.75       624
          25       0.71      0.76      0.73       563

    accuracy                           0.76     16000
   macro avg       0.76      0.75      0.76     16000
weighted avg       0.76      0.76      0.76     16000
```

Slika 18: Standardni izlaz koji ispisuje classification_report za predvidjanja nad skupom za obučavanje naspram skupa za obučavanje

	precision	recall	f1-score	support
0	0.88	0.86	0.87	146
1	0.63	0.83	0.72	159
2	0.76	0.77	0.76	133
3	0.70	0.82	0.75	175
4	0.67	0.73	0.70	147
5	0.79	0.72	0.75	155
6	0.58	0.52	0.55	147
7	0.74	0.50	0.59	171
8	0.79	0.84	0.81	132
9	0.94	0.77	0.85	151
10	0.70	0.76	0.73	147
11	0.94	0.73	0.82	161
12	0.78	0.92	0.84	160
13	0.91	0.72	0.81	147
14	0.75	0.71	0.73	154
15	0.92	0.81	0.86	173
16	0.80	0.68	0.74	149
17	0.58	0.74	0.65	139
18	0.49	0.59	0.53	148
19	0.81	0.82	0.81	157
20	0.88	0.84	0.86	176
21	0.78	0.80	0.79	160
22	0.76	0.92	0.84	133
23	0.76	0.82	0.79	147
24	0.78	0.74	0.76	162
25	0.81	0.78	0.79	171
accuracy			0.76	4000
macro avg	0.77	0.76	0.76	4000
weighted avg	0.77	0.76	0.76	4000

Slika 19: classification_report nad test skupom

Ovde na slici 19. izvodi se proračun nad predviđanjem skupom za testiranje modela za svaku vrednost klasifikatornog atributa, implementiranog po implementaciji 12.

Slova L(11.), P(15.), N(13.) imaju visoku preciznost. M(12.), X(22.), A(0.) imaju visoku senzitivnost. F1-score A(0.), P(15.), U(20.). Manja je zastupljenost nego ranije zbog manjeg broja (4000 njih) sagledanih zastupljenih instanci.

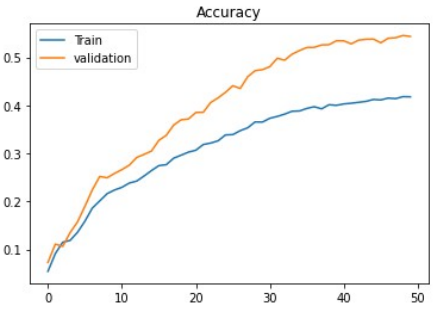
Statističke vrednosti metrika tačnosti, macro-avg, weighted-avg su slične.

```
#...
model = KerasClassifier(build_fn = baseline_model,
                                epochs = 2000,
                                batch_size = 256,
                                verbose = 1)
optimizers = ['Nadam']
activations = ['tanh']
init = ['normal']
param_grid = {
    'units': [15],
    'init': init,
    'activation': activations,
    'optimizer' : optimizers
}

grid = GridSearchCV(estimator=model,
                            param_grid=param_grid,
                            cv= 2)
grid_result = grid.fit(x_train, y_train,
                           validation_data = (x_test, y_test))
print("Best Parameters: ", grid_result.best_params_)
print("Best Score: ", grid_result.best_score_)

history = grid.best_estimator_.model.history

plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Accuracy')
plt.legend(['Train', 'validation'])
plt.show()
#... isto tako i za loss...

predict_train = grid_result.predict(x_train)
print(classification_report(y_train.argmax(axis=1), predict_train))

cm_train = confusion_matrix(y_train.argmax(axis=1), predict_train)

fig, ax = plt.subplots(figsize=(26,26))
ax = sns.heatmap(cm_train, annot = True, cmap = 'Blues', cbar=False, linewidths=.5, ax=ax)
ax.set_xlabel('Predicted Values')
ax.set_ylabel('Actual Values')
plt.show()
# ...isto tako i za test skup podataka...
```

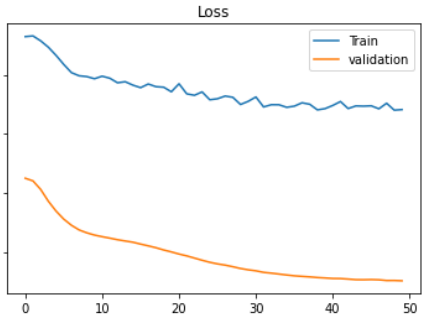
Implementacija 12: Deo koda funkcije evaluate_statistic_metrics_custom(x,y) *nad kojim se vršila ova pismena analiza*

5. Izvršavanje optimizacije hiperparametara neuronske mreže

Po podešavanju hiperparametara po postavci iz 3. sekcije u ovom radu. Izvršavana su upoređivanja metoda optimizacija 'SGD', 'RMSprop', 'Adagrad', 'Adadelta', 'Adam', 'Adamax', 'Nadam', aktivacionih funkcija 'softmax', 'softplus', 'softsign', 'relu', 'tanh', 'sigmoid', 'hard_sigmoid', 'linear', brojnosti neurona [5, 10, 15] u ulaznom i skrivenom sloju. Broj epoha je manji nego u prethodnoj 4. sekciji rada, a umesto stratifikovanog cross-validation-a se koristi "istumbani" k-fold. Tačnost sračunata po epohi za najbolji set hiperparametara nakon optimizacije je prikazana na slici 20., kao i gubitak na slici 21.



Slika 20: Tačnost sračunata po epohi za najbolji set hiperparametara nakon optimizacije



Slika 21: Gubitak sračunat po epohi za najbolji set hiperparametara nakon optimizacije

Izvedene metrike i statističke vrednosti nakon predvidjanja nad skupom podataka namenjenog za obučavanje po feature vrednostima je prikazana na slici 22, a nad skupom podataka namenjenog za testiranje na slici 23. Izlaz standardnog izlaza nakon evaluiranja posle svih epoha sa informacijama o odabranom najboljem skupu hiperparametara i ocenom prikazan u odeljku izlaz 1.

	precision	recall	f1-score	support
0	0.86	0.80	0.83	643
1	0.47	0.54	0.50	607
2	0.59	0.61	0.60	603
3	0.49	0.54	0.51	630
4	0.44	0.42	0.43	621
5	0.45	0.42	0.44	620
6	0.49	0.50	0.49	626
7	0.61	0.34	0.44	563
8	0.67	0.72	0.70	623
9	0.50	0.52	0.51	596
10	0.50	0.53	0.51	592
11	0.76	0.71	0.73	600
12	0.70	0.82	0.76	632
13	0.56	0.53	0.54	636
14	0.40	0.37	0.38	599
15	0.63	0.68	0.65	630
16	0.32	0.17	0.22	634
17	0.45	0.42	0.43	619
18	0.37	0.36	0.37	600
19	0.32	0.27	0.29	639
20	0.64	0.81	0.72	637
21	0.76	0.82	0.79	604
22	0.79	0.84	0.82	619
23	0.33	0.42	0.37	640
24	0.46	0.47	0.47	624
25	0.66	0.66	0.66	563
accuracy			0.55	16000
macro avg	0.55	0.55	0.54	16000
weighted avg	0.55	0.55	0.54	16000

Slika 22: Izvedene metrike i statističke vrednosti nakon predvidjanja nad skupom podataka namenjenog za obučavanje po feature vrednostima.

	precision	recall	f1-score	support
0	0.87	0.80	0.83	146
1	0.56	0.52	0.54	159
2	0.60	0.63	0.61	133
3	0.49	0.55	0.52	175
4	0.35	0.40	0.37	147
5	0.44	0.46	0.45	155
6	0.41	0.47	0.44	147
7	0.53	0.27	0.36	171
8	0.64	0.77	0.70	132
9	0.57	0.55	0.56	151
10	0.48	0.52	0.50	147
11	0.73	0.65	0.69	161
12	0.65	0.82	0.72	160
13	0.53	0.47	0.50	147
14	0.43	0.36	0.39	154
15	0.65	0.65	0.65	173
16	0.33	0.18	0.23	149
17	0.35	0.38	0.37	139
18	0.32	0.32	0.32	148
19	0.36	0.29	0.32	157
20	0.65	0.83	0.73	176
21	0.71	0.76	0.74	160
22	0.81	0.90	0.85	133
23	0.37	0.46	0.41	147
24	0.47	0.48	0.47	162
25	0.69	0.65	0.67	171
accuracy			0.54	4000
macro avg	0.54	0.54	0.54	4000
weighted avg	0.54	0.54	0.54	4000

Slika 23: Izvedene metrike i statističke vrednosti nakon predvidjanja nad skupom podataka namenjenog za testiranje po feature vrednostima.

```
1 Best Parameters:  {'activation': 'linear', 'init': 'normal', 'optimizer': 'RMSprop', 'units': 15}
2 Best Score:  0.5295631090799967
```

Izlaz 1: Izlaz nakon evaluiranja posle svih epoha sa informacijama o odabranom najboljem skupu hiperparametara i ocenom

6. Pojašnjenje prednosti i mana ovih metoda

Kroz ovaj rad uviđa se drastično gori performans modela koji je bio podložan optimizaciji hiperparametara što im ubija smisao za njihovu svrhu. Najviše se odazire ova problematika u tome što je korišćen stratifikacioni fold sa 2 cross-validation-a kao i veći broj epoha, a pri optimizaciji hiperparametara je korišćen shuffled-ovani k-fold sa 3 cross-validation-a.

Postavljeni kandidati hiperparametri pre optimizacije za optimizator su postavljeni bili tako da svaki optimizator je sledeće rešenje onih prethodnih ('SGD' na 'RMSprop'; 'Adagrad', 'Adadelta', 'Adam', 'Adamax' na 'Nadam'). Za kandidate aktivacionih funkcija iznosi se slično sagledanje (na primer. sigmoid na hardsigmoid).

Linearna aktivaciona funkcija je korišćena da se sagleda na skrivenom sloju iako to nije preporučljivo.

Takođe nije preporučljivo imati isti broj neurona na ulaznom sloju kao i u 1. skrivenom sloju.

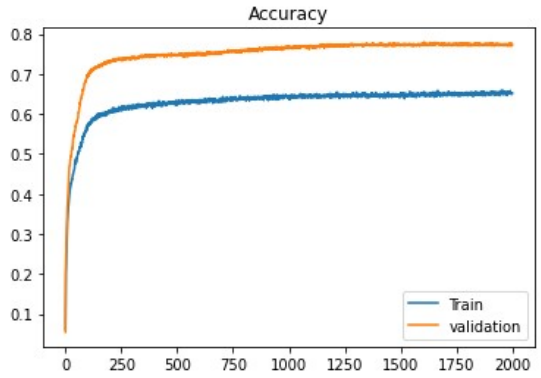
Analiziranja metrikama koretnosti predikcija naspram modela neuronske mreže da nakon .fit(...) funkcije i postavljene validacije trening skupa nad test skupom, da test skup ima bolje poklapanje tačnosti naspram modela.

Malo bolji su postignuti rezultati sa izmenom prikazanih na implementaciji 13., gde je uklonjen model.Dropout(...) na zadnjem sloju neurona.

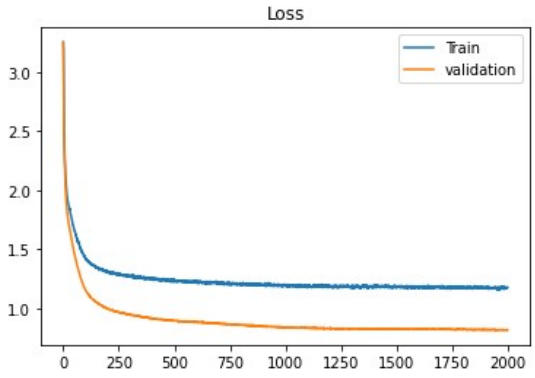
```
def baseline_model(optimizer='Nadam', units=10, init='uniform', activation='relu'):
        model = Sequential()
        model.add(Dense(units, input_dim = 16, kernel_initializer=init, activation = activation))
        model.add(Dropout(0.2))
        model.add(Dense(units, kernel_initializer=init, activation = activation))
        model.add(Dropout(0.2))
        model.add(Dense(26, kernel_initializer=init, activation = 'softmax'))
        model.compile(loss = 'categorical_crossentropy',
                        optimizer = optimizer,
                                metrics = ['accuracy'])
        return model
```

Implementacija 13: Popravka procesa overfitting-a nad modelom višeslojne neuronske mreže

Na slikama 24., 25. mogu da se vide posledice popravke pri evaluate_statistic_metrics_custom(x,y). Gde ostaje ista problematika, ali sa manjim odstupanjima. Statističke metrike i statističke analaze proračuna nad skupom za obučavanje po predikciji na slici 26., i nad skupom za testiranje po predikciji na slici 27.



Slika 24: Tačnost nakon evaluacija sa evaluate_statistic_metrics_custom(x,y)



Slika 25: Gubitak nakon evaluacija sa evaluate_statistic_metrics_custom(x,y)

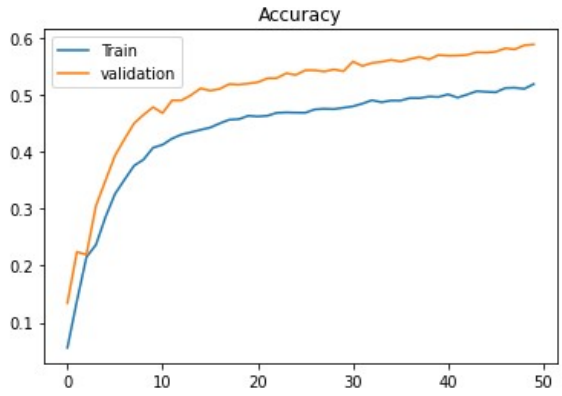
	precision	recall	f1-score	support
0	0.89	0.89	0.89	643
1	0.64	0.75	0.69	607
2	0.82	0.78	0.80	603
3	0.77	0.80	0.78	630
4	0.79	0.76	0.77	621
5	0.79	0.68	0.73	620
6	0.65	0.68	0.67	626
7	0.61	0.40	0.48	563
8	0.81	0.83	0.82	623
9	0.91	0.81	0.86	596
10	0.70	0.71	0.70	592
11	0.91	0.82	0.86	600
12	0.81	0.88	0.84	632
13	0.83	0.77	0.80	636
14	0.67	0.77	0.71	599
15	0.84	0.83	0.84	630
16	0.80	0.68	0.73	634
17	0.64	0.79	0.71	619
18	0.58	0.52	0.55	600
19	0.83	0.78	0.81	639
20	0.84	0.89	0.86	637
21	0.84	0.87	0.85	604
22	0.78	0.88	0.83	619
23	0.78	0.83	0.80	640
24	0.80	0.79	0.79	624
25	0.73	0.80	0.76	563
accuracy			0.77	16000
macro avg	0.77	0.77	0.77	16000
weighted avg	0.77	0.77	0.77	16000

Slika 27: Statističke metrike i statističke analize proracuna pri predikciji nad skupom podataka za obučavanje

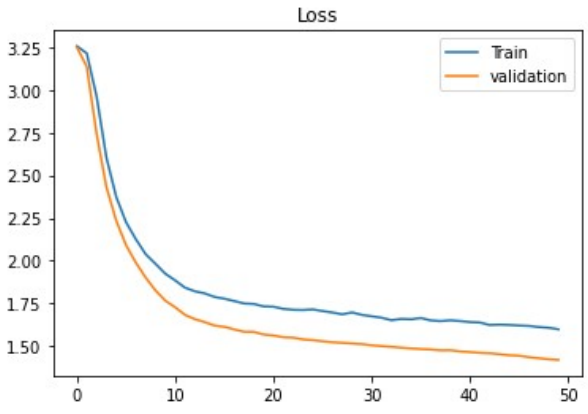
	precision	recall	f1-score	support
0	0.91	0.88	0.89	146
1	0.69	0.77	0.72	159
2	0.81	0.79	0.80	133
3	0.77	0.77	0.77	175
4	0.74	0.74	0.74	147
5	0.76	0.69	0.73	155
6	0.57	0.62	0.59	147
7	0.63	0.42	0.50	171
8	0.75	0.83	0.79	132
9	0.92	0.85	0.89	151
10	0.66	0.73	0.69	147
11	0.86	0.76	0.81	161
12	0.82	0.90	0.86	160
13	0.84	0.83	0.84	147
14	0.71	0.77	0.74	154
15	0.88	0.82	0.85	173
16	0.83	0.69	0.75	149
17	0.59	0.74	0.66	139
18	0.62	0.53	0.57	148
19	0.87	0.83	0.85	157
20	0.87	0.88	0.88	176
21	0.84	0.84	0.84	160
22	0.74	0.89	0.81	133
23	0.80	0.88	0.84	147
24	0.81	0.85	0.83	162
25	0.80	0.81	0.81	171
accuracy			0.77	4000
macro avg	0.77	0.77	0.77	4000
weighted avg	0.77	0.77	0.77	4000

Slika 26: Statističke metrike i statističke analize proracuna pri predikciji nad skupom podataka za testiranje

Takođe, obavljen je pregled i za evaluaciju uz optimizaciju nad hiperparametrima. Tačnost nakon evaluacija uz optimizaciju hiperparametara kroz 50 epoha i 3 kFold cross-validation-a prikazan je na slici 28., kao i gubitak na slici 29.



Slika 29: Tačnost nakon evaluacija uz optimizaciju hiperparametara



Slika 28: Gubitak nakon evaluacija uz optimizaciju hiperparametara

Statističke metrike i statističke analize proracuna pri predikciji nad skupom podataka za obučavanje prikazan na slici 30., kao i nad skupom podataka za testiranje na slici 31.

	precision	recall	f1-score	support
0	0.81	0.84	0.82	643
1	0.56	0.58	0.57	607
2	0.67	0.65	0.66	603
3	0.59	0.54	0.57	630
4	0.50	0.41	0.45	621
5	0.49	0.42	0.45	620
6	0.56	0.49	0.52	626
7	0.51	0.40	0.45	563
8	0.70	0.75	0.72	623
9	0.57	0.66	0.61	596
10	0.43	0.46	0.44	592
11	0.76	0.76	0.76	600
12	0.68	0.84	0.75	632
13	0.55	0.54	0.54	636
14	0.47	0.44	0.45	599
15	0.62	0.69	0.65	630
16	0.50	0.62	0.55	634
17	0.57	0.68	0.62	619
18	0.37	0.33	0.35	600
19	0.44	0.33	0.38	639
20	0.78	0.79	0.78	637
21	0.74	0.85	0.79	604
22	0.82	0.86	0.84	619
23	0.41	0.38	0.39	640
24	0.54	0.51	0.52	624
25	0.70	0.71	0.71	563
accuracy			0.60	16000
macro avg	0.59	0.60	0.59	16000
weighted avg	0.59	0.60	0.59	16000

Slika 30: Statističke metrike i statističke analize proracuna pri predikciji nad skupom podataka za obučavanje

	precision	recall	f1-score	support
0	0.81	0.83	0.82	146
1	0.62	0.54	0.58	159
2	0.67	0.64	0.66	133
3	0.68	0.55	0.61	175
4	0.40	0.38	0.39	147
5	0.47	0.49	0.48	155
6	0.50	0.45	0.47	147
7	0.53	0.37	0.44	171
8	0.64	0.77	0.70	132
9	0.60	0.68	0.64	151
10	0.43	0.48	0.45	147
11	0.72	0.68	0.70	161
12	0.63	0.83	0.72	160
13	0.50	0.48	0.49	147
14	0.46	0.44	0.45	154
15	0.67	0.66	0.67	173
16	0.48	0.61	0.54	149
17	0.52	0.66	0.58	139
18	0.29	0.28	0.29	148
19	0.49	0.36	0.41	157
20	0.77	0.80	0.78	176
21	0.70	0.81	0.75	160
22	0.83	0.91	0.87	133
23	0.44	0.41	0.43	147
24	0.57	0.51	0.54	162
25	0.73	0.68	0.70	171
accuracy			0.59	4000
macro avg	0.58	0.59	0.58	4000
weighted avg	0.59	0.59	0.58	4000

Slika 31: Statističke metrike i statističke analize proracuna pri predikciji nad skupom podataka za testiranje

Izlaz standardnog izlaza nakon evaluiranja posle svih epoha sa informacijama o odabranom najboljem skupu hiperparametara i ocenom prikazan u odeljku izlaz 2.

```
Best Parameters:  {'activation': 'linear', 'init': 'normal', 'optimizer': 'Adam', 'units': 15}

Best Score:  0.5489998261133829
```

Izlaz 2: Odabir najboljih parametara pri optimizaciji hiperparametra

Predost je u svemu ovome što je omogućeno efikasno ažuriranje stanja neurona, propagacija kroz mrežu, boljeg načina pri obučavanju pomoću načina konstrukcija podela(ušteda memorije), predikcija, a da se ne vrše nepotrebna ponavljanja, dat neki faktor nasumičnosti, kao i ponavljanja nasumičnosti algoritama iznova.

Bibliografija

1: KEEL-dataset - data set description, https://sci2s.ugr.es/keel/dataset.php?cod=198
2: Normalized histogram statistics, https://seaborn.pydata.org/tutorial/distributions.html#normalized-histogram-statistics
3: How to make a color map with many unique colors in seaborn, https://stackoverflow.com/questions/68209351/how-to-make-a-color-map-with-many-unique-colors-in-seaborn
4: How do I choose the optimal batch size?, https://ai.stackexchange.com/questions/8560/how-do-i-choose-the-optimal-batch-size
5: Various Optimization Algorithms For Training Neural Network, https://towardsdatascience.com/optimizers-for-training-neural-network-59450d71caf6
6: Keras layer initializers, https://keras.io/api/layers/initializers/
7: What are the advantages of ReLU over sigmoid function in deep neural networks?, https://stats.stackexchange.com/questions/126238/what-are-the-advantages-of-relu-over-sigmoid-function-in-deep-neural-networks
8: Softmax Activation Function with Python, https://machinelearningmastery.com/softmax-activation-function-with-python/
9: What Is Cross Entropy In Python?, https://www.askpython.com/python-modules/numpy/cross-entropy-in-python
10: A Look at Gradient Descent and RMSprop Optimizers, https://towardsdatascience.com/a-look-at-gradient-descent-and-rmsprop-optimizers-f77d483ef08b
11: Deep Learning Optimizers, https://towardsdatascience.com/deep-learning-optimizers-436171c9e23f
12: 10 famous Machine Learning Optimizers , https://dev.to/amananandrai/10-famous-machine-learning-optimizers-1e22
13: Deep Learning Course — Lesson 7.5: Nadam (Nesterov-accelerated Adaptive Moment Estimation), https://medium.com/@nerdjock/deep-learning-course-lesson-7-5-nadam-nesterov-accelerated-adaptive-moment-estimation-efe9050d5b9b
14: What is the tanh activation function?, https://www.educative.io/answers/what-is-the-tanh-activation-function
15: Python implementation of Activation function for Neural Network, https://tariqueakhtar-39220.medium.com/python-implementation-of-activation-function-for-neural-network-cb38fe735c9e
16: Deep study of a not very deep neural network. Part 2: Activation functions, https://towardsdatascience.com/deep-study-of-a-not-very-deep-neural-network-part-2-activation-functions-fd9bd8d406fc
17: Accuracy Score:GridSearchCV, https://scikit-learn.org/stable/modules/model_evaluation.html#accuracy-score
18: How to Choose Loss Functions When Training Deep Learning Neural Networks, https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/
19: Machine Learning Model Evaluation Metrics part 2: Multi-class classification, https://www.mariakhalusova.com/posts/2019-04-17-ml-model-evaluation-metrics-p2/
20: Classification metrics: Multiclass and multilabel classification, https://scikit-learn.org/stable/modules/model_evaluation.html#multiclass-and-multilabel-classification