

Ministerul Educației și Cercetării al Republicii Moldova

Universitatea Tehnică a Moldovei

Facultatea Calculatoare, Informatică și Microelectronică

Departamentul Ingineria Software și Automatică

**RAPORT**

**Lucrarea de laborator nr.4**

**la Inteligența Artificială**

*Tema: Rețele neuronale artificiale.*

Grupa academică: TI-214  
A efectuat: Reguș Ruslan

**A verificat: Mariana Rusu**

Chișinău 2024

**Scopul lucrării:**

De realizat scripturile din cadrul sarcinilor pentru studierea modului de creare a unei rețele neuronale artificiale.

**Sarcina lucrării:**

De creat o rețea neuronală artificială în limbajul Python.

O rețea neuronală artificială este un model matematic inspirat din sistemul nervos biologic, utilizat pentru a rezolva probleme complexe de învățare automată și inteligență artificială. Aceasta este compusă dintr-o serie de straturi de neuroni interconectați, fiecare fiind o unitate de bază de procesare a informațiilor.

Componentele unei rețele neuronale artificiale:

* Straturi de neuroni (input, hidden, output);
* Funcții de activare (sigmoid, relu, tanh etc.);
* Funcții de cost (pentru evaluarea performanței modelului);
* Optimizatori (gradient descent, Adam, RMSprop etc.).

RNA:

import pandas as pd

import numpy as np

import os

os.environ['TF\_ENABLE\_ONEDNN\_OPTS'] = '0'

np.random.seed(42)

# Generăm un set de date fictiv despre performanța studenților la un examen universitar

n\_samples = 1000

exam\_score = np.random.randint(0, 101, n\_samples) # Scorurile sunt în intervalul [0, 100]

hours\_studied = np.random.uniform(0, 10, n\_samples) # Numărul de ore studiate

attendance = np.random.randint(0, 2, n\_samples) # Prezența la cursuri (0 - absent, 1 - prezent)

grades = np.random.randint(5, 11, n\_samples) # Notele finale (pe o scară de la 5 la 10)

pass\_exam = ((exam\_score >= 50) & (grades >= 6)).astype(int) # Dacă un student a promovat examenul (1 - da, 0 - nu)

data = pd.DataFrame({

'ExamScore': exam\_score,

'HoursStudied': hours\_studied,

'Attendance': attendance,

'Grades': grades,

'PassExam': pass\_exam

})

print(data.head())

# Separăm variabila țintă (PassExam) de restul datelor

X = data.drop('PassExam', axis=1)

y = data['PassExam']

# Divizăm datele în seturi de antrenare și testare

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Scalarea datelor

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

# Construim modelul rețelei neuronale

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Dropout

model = Sequential([

Dense(64, activation='relu', input\_dim=X\_train\_scaled.shape[1]),

Dropout(0.2),

Dense(32, activation='relu'),

Dropout(0.2),

Dense(1, activation='sigmoid')

])

model.compile(optimizer='adam',

loss='binary\_crossentropy',

metrics=['accuracy'])

# Antrenăm modelul și obținem istoricul antrenării

history = model.fit(X\_train\_scaled, y\_train, epochs=30, batch\_size=10, validation\_split=0.1, verbose=0)

# Evaluăm performanța modelului pe setul de testare

loss, accuracy = model.evaluate(X\_test\_scaled, y\_test, verbose=0)

print(f'Loss pe setul de testare: {loss:.4f}')

print(f'Acuratețe pe setul de testare: {accuracy:.4f}')

# Analizăm evoluția pierderilor și a acurateței în timpul antrenării

import matplotlib.pyplot as plt

plt.figure(figsize=(10, 5))

# Evoluția pierderilor

plt.subplot(1, 2, 1)

plt.plot(history.history['loss'], label='Loss (Antrenare)')

plt.plot(history.history['val\_loss'], label='Loss (Validare)')

plt.title('Evoluția pierderilor în timpul antrenării')

plt.xlabel('Epocă')

plt.ylabel('Pierdere')

plt.legend()

# Evoluția acurateței

plt.subplot(1, 2, 2)

plt.plot(history.history['accuracy'], label='Acuratețe (Antrenare)')

plt.plot(history.history['val\_accuracy'], label='Acuratețe (Validare)')

plt.title('Evoluția acurateței în timpul antrenării')

plt.xlabel('Epocă')

plt.ylabel('Acuratețe')

plt.legend()

plt.tight\_layout()

plt.show()

Scopul rețelei neuronale prezentate este de a prezice dacă un student a promovat examenul universitar pe baza unor caracteristici cum ar fi scorul la examen, numărul de ore studiate, prezența la cursuri și notele finale. Aceasta utilizează o arhitectură simplă cu trei straturi neuronale: un strat de intrare cu dimensiunea specificată de caracteristici, două straturi ascunse cu funcții de activare ReLU(straturi de procesare a informației) și un strat de ieșire cu funcția de activare sigmoid pentru a prezice o probabilitate între 0 și 1 pentru promovare. De asemenea, modelul aplică tehnici de regularizare prin dropout pentru a evita suprastudența(posibilitatea de supraantrenare a modelului încât acesta nu se mai bazează pe caracteristicile generale ale problemei) și îmbunătățirea generalizării.

**Arhitectura rețelei neuronale:**

* Modelul folosește o arhitectură simplă cu un strat de intrare, două straturi ascunse și un strat de ieșire.
* Funcții de activare ReLU sunt folosite pentru straturile ascunse, iar funcția de activare sigmoid este utilizată pentru stratul de ieșire pentru a prezice o probabilitate între 0 și 1 pentru trecerea examenului.

**Regularizare și dropout:**

* Dropout-ul este aplicat pentru a preveni suprastudența și pentru a îmbunătăți generalizarea modelului.
* Un dropout de 20% este aplicat după primul și al doilea strat dens.

**Funcția de pierdere și optimizatorul:**

* Pentru această problemă de clasificare binară, funcția de pierdere utilizată este binary\_crossentropy, iar optimizatorul este Adam.

**Evaluarea performanței:**

* Performanța modelului este evaluată pe setul de testare folosind metrici precum pierderea și acuratețea.

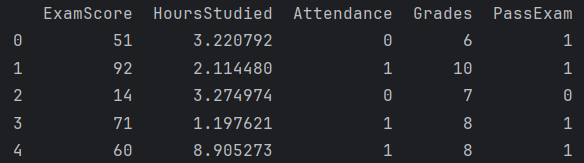


Figura 1 – Structura setului de date generat



Figura 2 – Calcularea metricilor pentru model

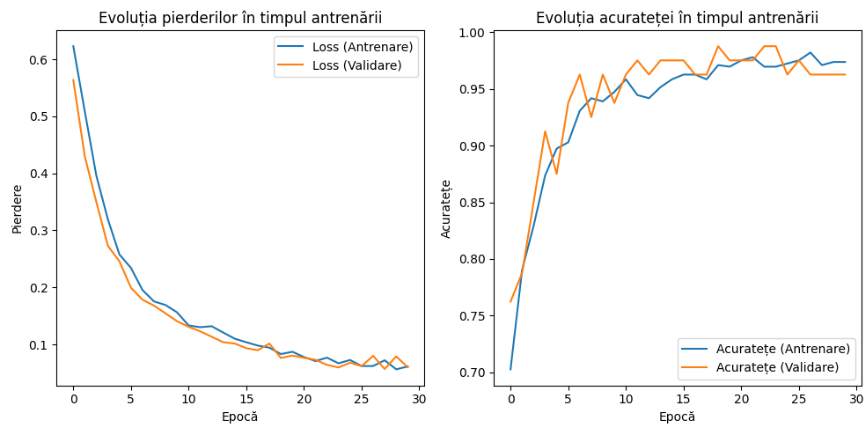


Figura 3 – Afișarea graficelor referitor la evoluția   
pierderilor și acurateței

Graficul evoluției pierderilor (loss) arată cum valoarea pierderii scade pe măsură ce modelul se antrenează, indicând că acesta învață să facă predicții mai precise. Pe de altă parte, graficul evoluției acurateții arată cum acuratețea modelului crește pe măsură ce acesta se ajustează la datele de antrenare și de validare, reflectând capacitatea sa de a face predicții corecte. Analizând aceste grafice, putem observa cum modelul se îmbunătățește în timpul procesului de antrenare și cât de bine se adaptează la datele din setul de antrenare și validare.

**Concluzie:**

În concluzie, analiza codului și rezultatele obținute arată o abordare eficientă în construirea și evaluarea unei rețele neuronale artificiale pentru a prezice promovarea unui examen universitar pe baza unor caracteristici relevante. Setul de date generat a fost adecvat pentru scopul propus, iar modelul rețelei neuronale, configurat cu straturi dens conectate și regularizare dropout, a obținut o performanță impresionantă în prezicerea rezultatului examenului. Graficele evoluției pierderilor și acurateții reflectă o îmbunătățire constantă a performanței modelului pe parcursul antrenării, sugerând abilitatea sa de a învăța și generaliza corect pe datele de antrenare și de validare. Evaluarea pe setul de testare a confirmat capacitatea modelului de a face predicții utile în practică. Aceste rezultate constituie un fundament solid pentru optimizarea și extinderea ulterioară a modelului pentru probleme similare sau mai complexe în domeniul învățării automate și analizei predictivă.