**Proiect**

**Rețele neuronale**

**Inteligența Artificială**

Student,

PARTENI ELENA

CTI, Anul III, Grupa 22C31B

**Tema aleasă:** Predicția bolilor cardiace. Condiții de sănătate bazate pe diverși indicatori medicali.

**Cod:**

import pandas as pd

import numpy as np

from google.colab import files

#upload files in linie comanda

setdate = files.upload()

print(setdate)

#citirea datelor din fisierul csv

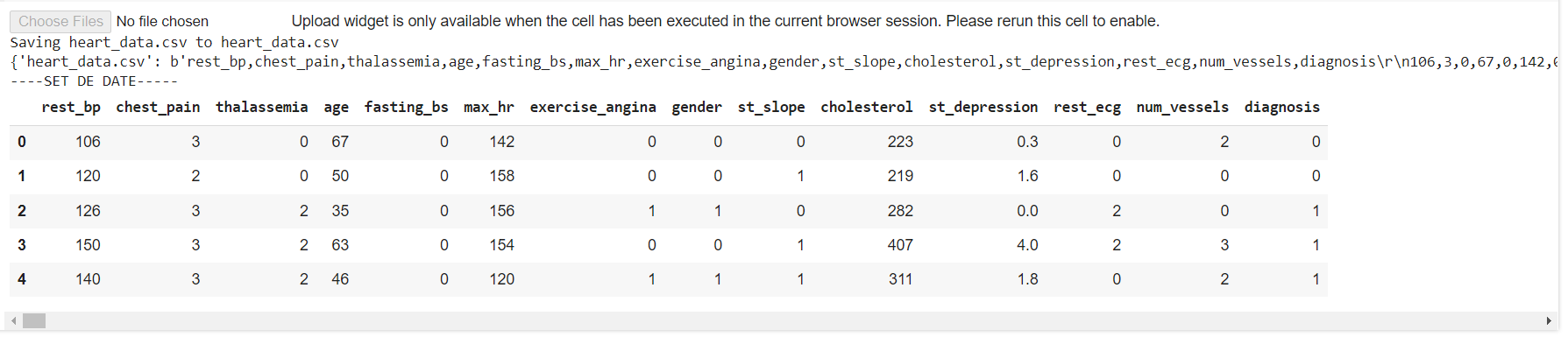
heart\_set = pd.read\_csv('heart\_data.csv')

#afisare primele randuri din heart\_set

heart\_array=np.array(heart\_set)

print("----SET DE DATE-----")

heart\_set.head()

****

//Calcularea range si standard deviation

# Coloanele din array-ul heart\_array alese

selected\_columns = [0, 1, 5, 9, 10]

# Calcularea și afișarea gama și deviația standard pentru fiecare coloană selectată

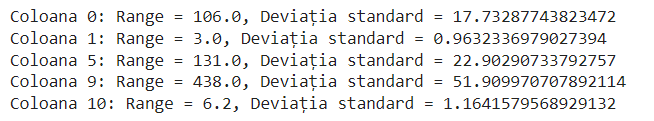
for column\_index in selected\_columns:

    column\_data = heart\_array[:, column\_index].astype(float)  # Convertim valorile în float pentru a evita erorile

    column\_range = np.ptp(column\_data) #calculează intervalul (diferența dintre maxim și minim) al valorilor din column\_data

    column\_std = np.std(column\_data) #calculează deviația standard a valorilor din column\_data

    print(f"Coloana {column\_index}: Range = {column\_range}, Deviația standard = {column\_std}")



//Histograms, box plots and scatter plots

import matplotlib.pyplot as plt

# Coloanele din array-ul heart\_array pentru care dorim să afisam graficele

selected\_columns = [0, 1, 5, 9, 10] #["rest\_bp", "chest\_pain", "max\_hr", "cholesterol", "st\_depression"]

# Setul de date

for column\_index in selected\_columns:

    column\_data = heart\_array[:, column\_index].astype(float)  # Convertim valorile în float pentru a evita erorile

    # Histograma

    plt.figure(figsize=(8, 6))

    plt.hist(column\_data, bins=10, color='skyblue', edgecolor='black')

    plt.title(f'Histograma pentru coloana {column\_index}')

    plt.xlabel('Valori')

    plt.ylabel('Frecvență')

    plt.show()

    # Graficul cu cutii (box plot)

    plt.figure(figsize=(8, 6))

    plt.boxplot(column\_data)

    plt.title(f'Graficul cu cutii pentru coloana {column\_index}')

    plt.ylabel('Valori')

    plt.show()

import matplotlib.pyplot as plt

# Coloanele din DataFrame-ul heart\_set pentru care dorim să afișăm graficele

selected\_columns = ["rest\_bp", "chest\_pain", "max\_hr", "cholesterol", "st\_depression"]

# Setul de date

for column\_name in selected\_columns:

    plt.figure(figsize=(8, 6))

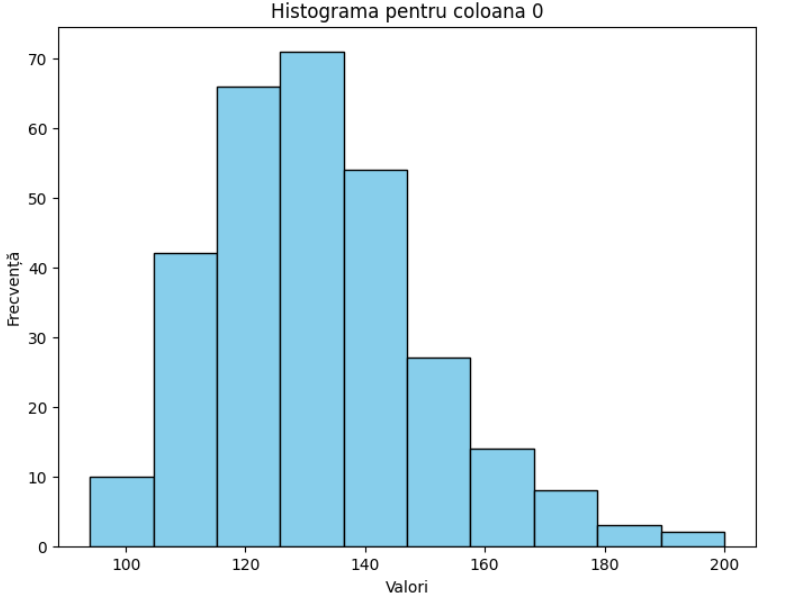
    plt.scatter(heart\_set.index, heart\_set[column\_name], color='green')

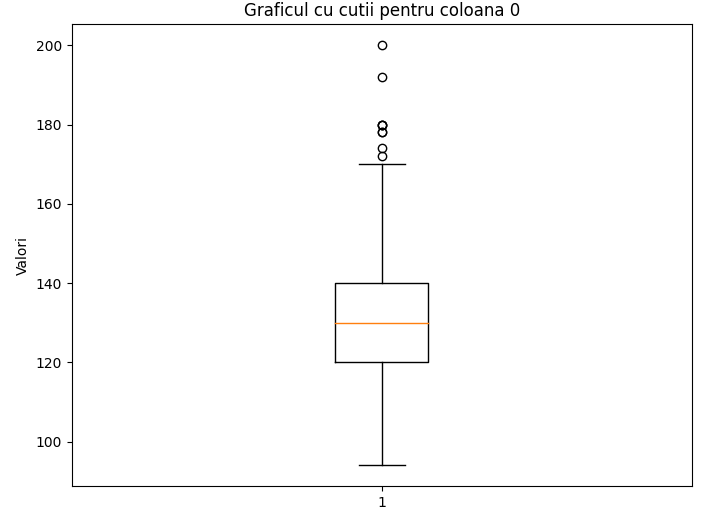
    plt.title(f'Scatter Plot of {column\_name} vs Index')

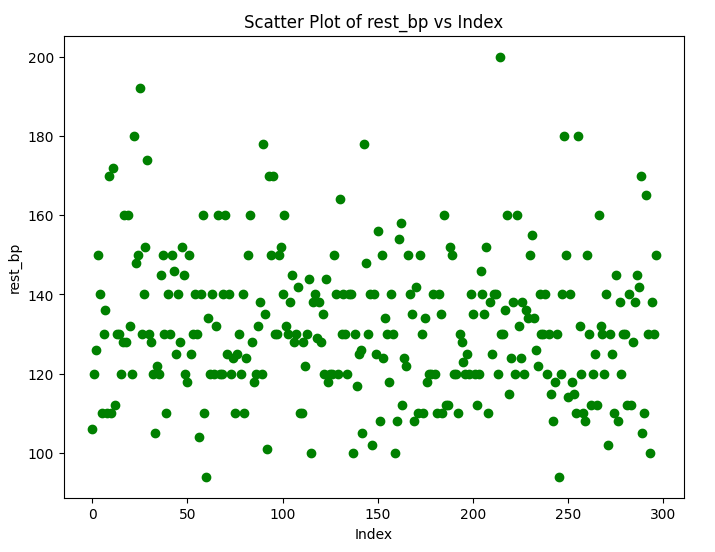
    plt.xlabel('Index')

    plt.ylabel(column\_name)

    plt.show()







//Matricea de corelații

# Selectam cel putin 5 coloane, inclusiv coloana țintă (diagnosis)

selected\_columns = ['rest\_bp', 'age', 'max\_hr', 'cholesterol', 'st\_depression', 'diagnosis']

# Calcularea matricei de corelatii

correlation\_matrix = heart\_set[selected\_columns].corr()

# Afisarea matricei de corelatii

print("Matricea de corelatii:")

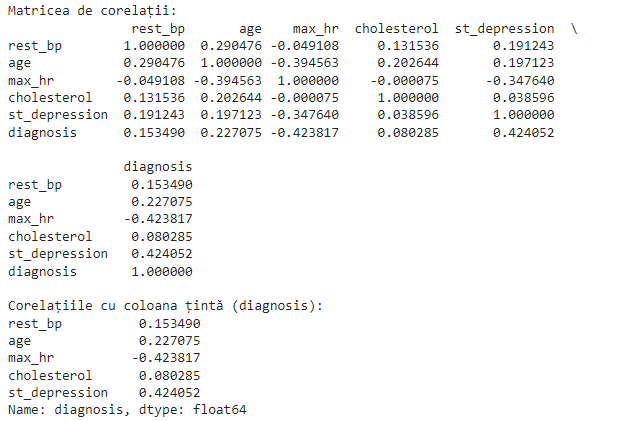
print(correlation\_matrix)

# Afisarea corelatiilor cu coloana tinta (diagnosis)

target\_correlations = correlation\_matrix['diagnosis'].drop('diagnosis')  # Eliminam corelatia cu ea insasi

print("\nCorelatiile cu coloana tinta (diagnosis):")

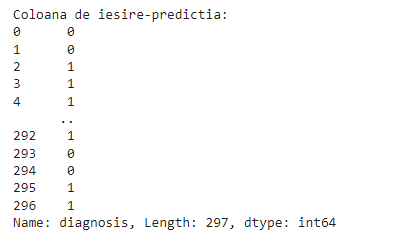
print(target\_correlations)



//Afișarea coloanei target

print("Coloana de iesire-predictia:")

print(heart\_set['diagnosis'])



//Impărțirea datelor in set de antrenare si set de testare

# impartirea setului de date:

# X\_train = intrari set de antrenare

# X\_test = iesire set de antrenare

# y\_train - intrari set de testare

# y\_test - iesiri set de testare

# test\_size=0.2: 20% setul de testare; 80% setul de antrenare

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

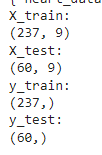
x=heart\_set.iloc[:,[0, 1, 2, 3, 4, 5, 6, 7, 10]].values

y=heart\_set.iloc[:,13].values

x\_train,x\_test, y\_train,y\_test = train\_test\_split(x,y,test\_size=0.2,random\_state=0, stratify=y)

print("X\_train:")

print(x\_train.shape)



print("X\_test:")

print(x\_test.shape)

print("y\_train:")

print(y\_train.shape)

print("y\_test:")

print(y\_test.shape)

//Definirea modelului neuronal

NR1.

#crearea modelului retelei neuronale

from keras.models import Sequential

from keras.layers import Dense

from keras.optimizers import SGD,Adam

import tensorflow as tf

model = Sequential()

# stratul de intrare:

# 1) numarul de neuroni =

# 2) functia de activare =

# 3) intrari = 8

# Adăugarea stratului de intrare

model.add(Dense(units=12, activation='relu', input\_dim=5))

#strat ascuns

model.add(tf.keras.layers.Dense(units=8, activation='relu'))

#strat de iesire

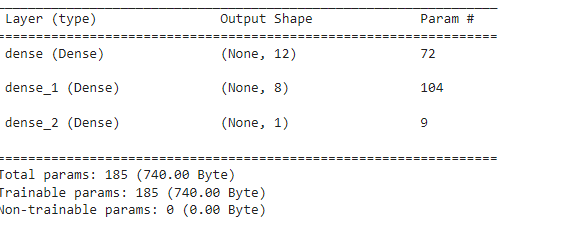
model.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))

# compilarea modelului cu functia de optimizare Adam; lr - rata de invatare; metrics = accuracy (acuratetea pentru cazul de iesiri binare)

model.compile(Adam(lr=0.04), 'binary\_crossentropy', metrics=['accuracy'])

# afisarea modelului

model.summary()



NR.2

from keras.models import Sequential

from keras.layers import Dense

from keras.optimizers import SGD,Adam

import tensorflow as tf

model = Sequential()

# stratul de intrare:

# 1) numarul de neuroni =

# 2) functia de activare =

# 3) intrari = 9

# Adăugarea stratului de intrare

model.add(Dense(units=12, activation='relu', input\_dim=9))

#strat ascuns

model.add(tf.keras.layers.Dense(units=8, activation='relu'))

#strat de iesire

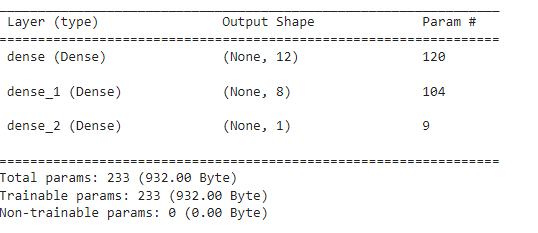
model.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))

# compilarea modelului cu functia de optimizare Adam; lr - rata de invatare; metrics = accuracy (acuratetea pentru cazul de iesiri binare)

model.compile(Adam(lr=0.04), 'binary\_crossentropy', metrics=['accuracy'])

# afisarea modelului

model.summary()



NR.3

#crearea modelului retelei neuronale

from keras.models import Sequential

from keras.layers import Dense

from keras.optimizers import SGD,Adam

import tensorflow as tf

model = Sequential()

# stratul de intrare:

# 1) numarul de neuroni =

# 2) functia de activare = tanh

# 3) intrari = 8

# Adăugarea stratului de intrare

model.add(Dense(units=20, activation='relu', input\_dim=9))

#strat ascuns

model.add(tf.keras.layers.Dense(units=12, activation='relu'))

#strat de iesire

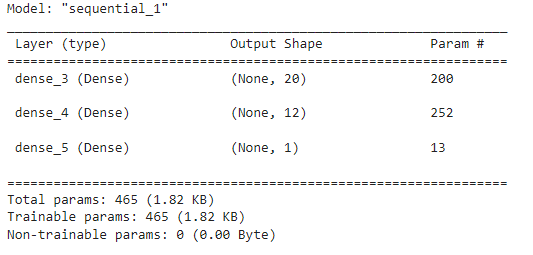
model.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))

# compilarea modelului cu functia de optimizare Adam; lr - rata de invatare; metrics = accuracy (acuratetea pentru cazul de iesiri binare)

model.compile(Adam(lr=0.04), 'binary\_crossentropy', metrics=['accuracy'])

# afisarea modelului

model.summary()



NR.4

#crearea modelului retelei neuronale

from keras.models import Sequential

from keras.layers import Dense

from keras.optimizers import SGD,Adam

import tensorflow as tf

model = Sequential()

# stratul de intrare:

# 1) numarul de neuroni =

# 2) functia de activare = tanh

# 3) intrari = 8

# Adăugarea stratului de intrare

model.add(Dense(units=16, activation='relu', input\_dim=9))

#strat ascuns

model.add(tf.keras.layers.Dense(units=12, activation='relu'))

#strat de iesire

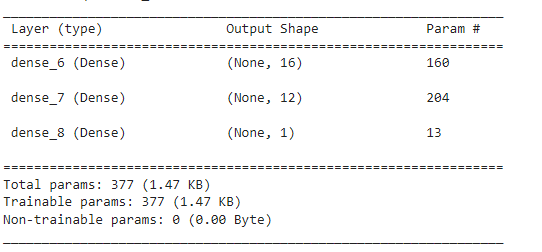
model.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))

# compilarea modelului cu functia de optimizare Adam; lr - rata de invatare; metrics = accuracy (acuratetea pentru cazul de iesiri binare)

model.compile(Adam(lr=0.04), 'binary\_crossentropy', metrics=['accuracy'])

# afisarea modelului

model.summary()



NR. 5

#crearea modelului retelei neuronale- NR. 5

from keras.models import Sequential

from keras.layers import Dense

from keras.optimizers import SGD,Adam

import tensorflow as tf

model = Sequential()

# stratul de intrare:

# 1) numarul de neuroni =

# 2) functia de activare =

# 3) intrari = 9

# Adăugarea stratului de intrare

model.add(Dense(units=16, activation='relu', input\_dim=9))

#strat ascuns

model.add(tf.keras.layers.Dense(units=14, activation='relu'))

#strat de iesire

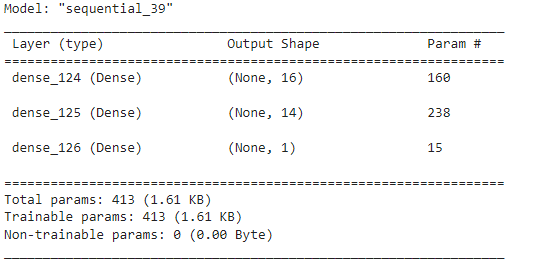
model.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))

# compilarea modelului cu functia de optimizare Adam; lr - rata de invatare; metrics = accuracy (acuratetea pentru cazul de iesiri binare)

model.compile(Adam(lr=0.04), 'binary\_crossentropy', metrics=['accuracy'])

# afisarea modelului

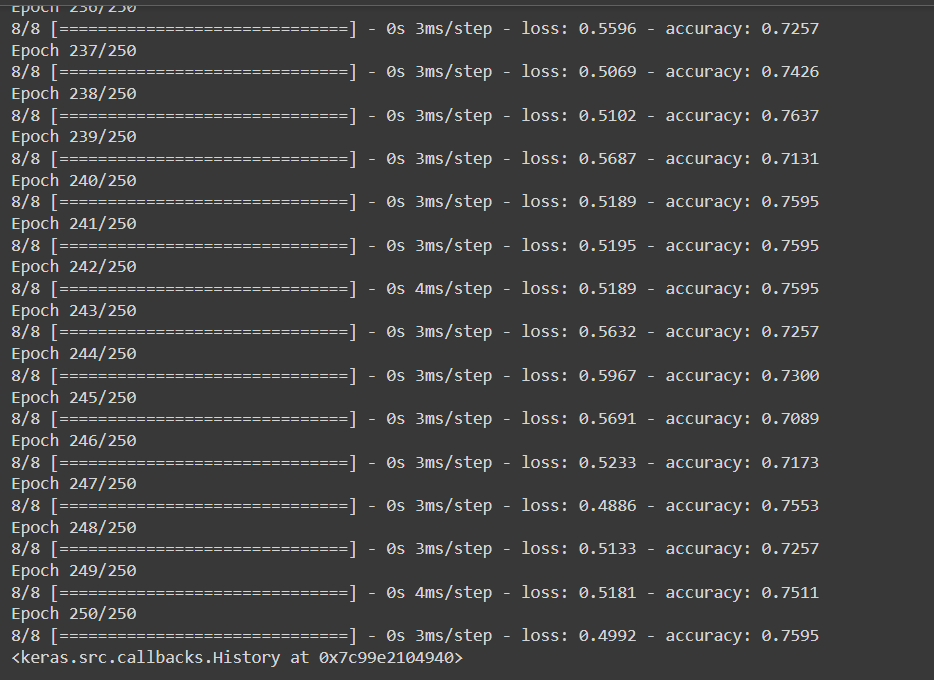
model.summary()



//Antrenarea modelului neuronal

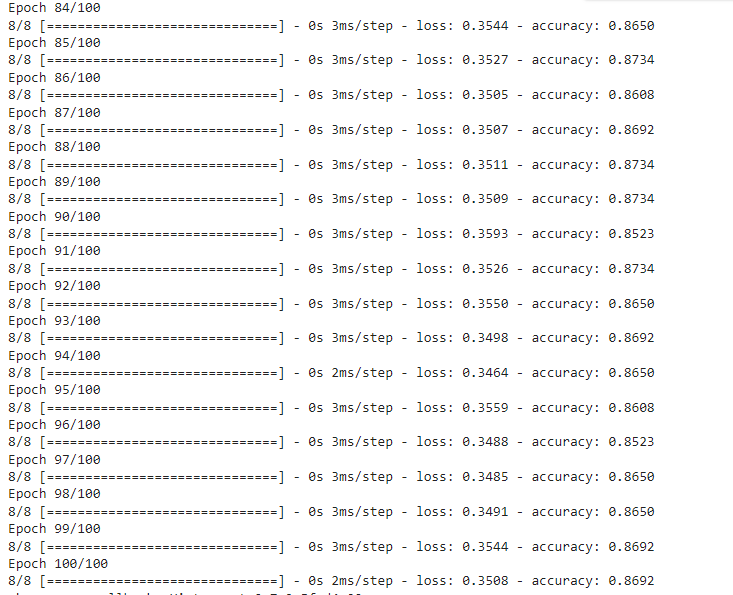
NR. 1

model.fit(x\_train,y\_train,epochs=250)



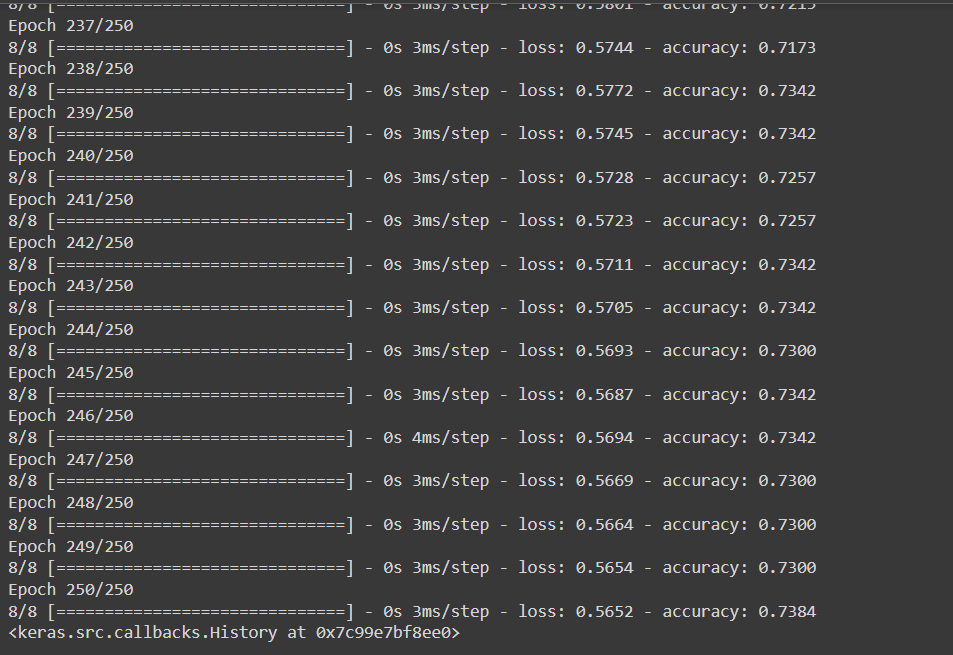
NR.2

model.fit(x\_train,y\_train,epochs=100)



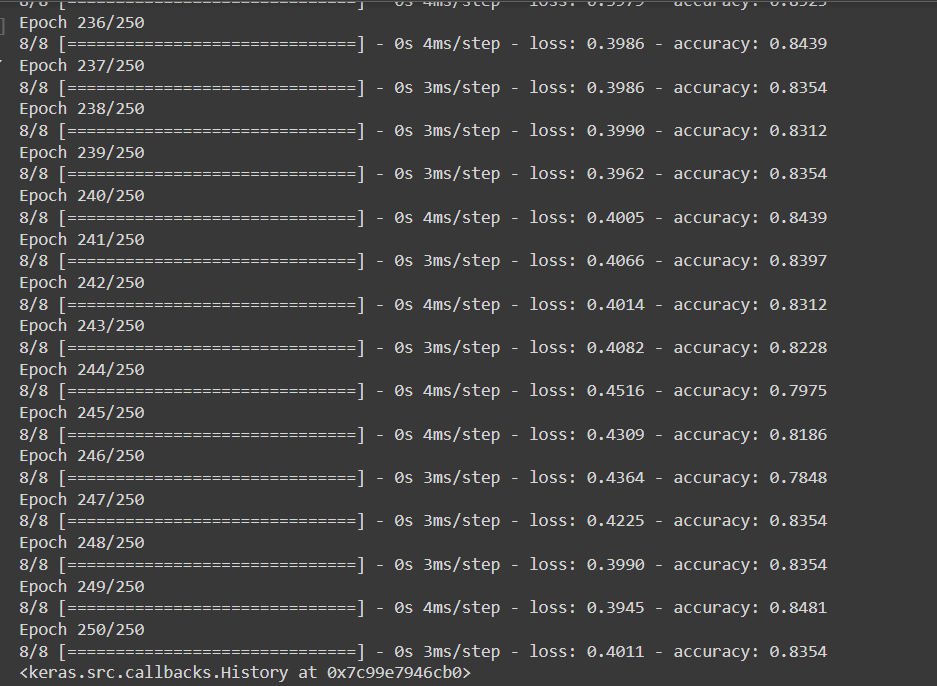
NR.3

model.fit(x\_train,y\_train,epochs=250)



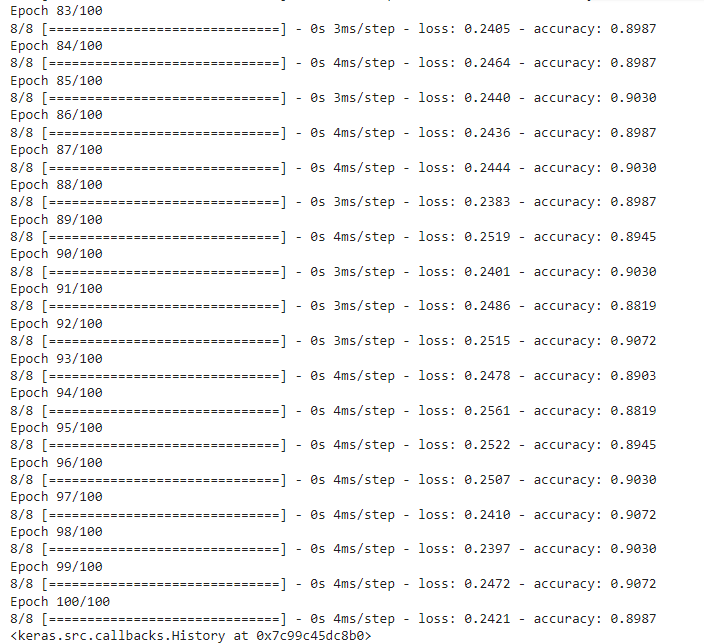
NR.4

model.fit(x\_train,y\_train,epochs=250)



NR. 5

model.fit(x\_train,y\_train,epochs=100)



//Acuratețea la antrenare si la testare

NR.2

# Calculul acurateței pe setul de antrenare

train\_loss, train\_accuracy = model.evaluate(x\_train, y\_train)

print("Acuratețea la antrenare:", train\_accuracy)

# Calculul acurateței pe setul de testare

test\_loss, test\_accuracy = model.evaluate(x\_test, y\_test)

print("Acuratețea la testare:", test\_accuracy)

8/8 [==============================] - 0s 3ms/step - loss: 0.3457 - accuracy: 0.8692

Acuratețea la antrenare: 0.8691983222961426

2/2 [==============================] - 0s 9ms/step - loss: 0.5966 - accuracy: 0.8000

Acuratețea la testare: 0.800000011920929

NR.5

# Calculul acurateței pe setul de antrenare

train\_loss, train\_accuracy = model.evaluate(x\_train, y\_train)

print("Acuratețea la antrenare:", train\_accuracy)

# Calculul acurateței pe setul de testare

test\_loss, test\_accuracy = model.evaluate(x\_test, y\_test)

print("Acuratețea la testare:", test\_accuracy)

# Generarea graficului pentru acuratețe

plt.figure(figsize=(12, 4))

# Acuratețea

plt.subplot(1, 2, 1)

plt.plot(history.history['accuracy'], label='Train Accuracy')

plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')

plt.title('Model Accuracy')

plt.xlabel('Epoch')

plt.ylabel('Accuracy')

plt.legend(loc='best')

# Pierderea

plt.subplot(1, 2, 2)

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

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

plt.title('Model Loss')

plt.xlabel('Epoch')

plt.ylabel('Loss')

plt.legend(loc='best')

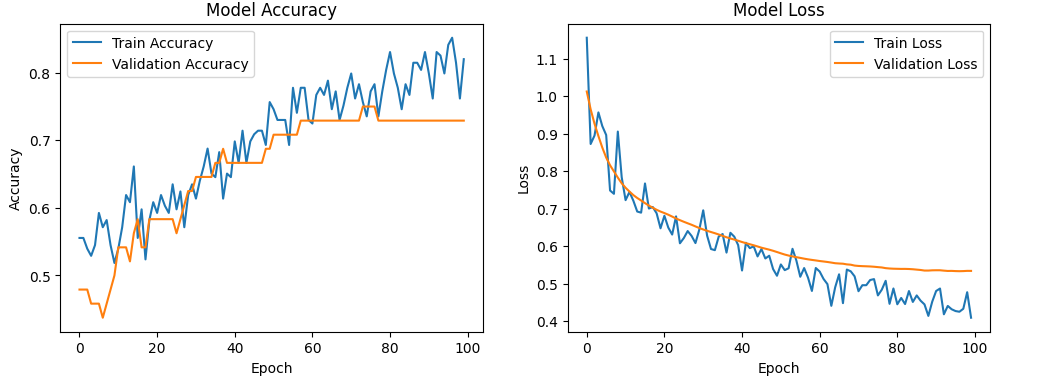
plt.show()

8/8 [==============================] - 0s 6ms/step - loss: 0.2338 - accuracy: 0.9030

Acuratețea la antrenare: 0.902953565120697

2/2 [==============================] - 0s 11ms/step - loss: 1.0766 - accuracy: 0.8333

Acuratețea la testare: 0.8333333134651184



|  |  |  |  |
| --- | --- | --- | --- |
| Nr. neuroni | Straturii | Functii | Observatii |
| 12,8,1 | 3 | Relu, relu, sigmoid | Am folosit coloanele 0,1,5,9,10 si am avut acuratete mica si loss mare |
| 12,8,1 | 3 | Relu, relu, sigmoid | Una dintre cele mai buna acuratete 0,8692 si am folosit coloanele de la 0-7 si 10 |
| 20,12,1 | 3 | Relu, relu, sigmoid | Am folosit coloanele de la 0-7 si 10, dar nu am avut rezultate asa de bune |
| 16,12,1 | 3 | Relu, relu, sigmoid | Am folosit coloanele de la 0-7 si 10, dar nu am avut rezultate asa de bune |
| 16,14,1 | 3 | Relu, relu, sigmoid | Am folosit coloanele de la 0-7 si 10, si am avut cea mai mare acuratete |