**Задача 1: Класификация на цифри с невронна мрежа (MNIST)**

from tensorflow.keras.datasets import mnist

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Flatten

from tensorflow.keras.utils import to\_categorical

import matplotlib.pyplot as plt

# Зареждане на MNIST

(X\_train, y\_train), (X\_test, y\_test) = mnist.load\_data()

# Нормализация

X\_train = X\_train / 255.0

X\_test = X\_test / 255.0

# Категоризация на изхода (one-hot encoding)

y\_train\_cat = to\_categorical(y\_train)

y\_test\_cat = to\_categorical(y\_test)

# Модел

model = Sequential([

Flatten(input\_shape=(28, 28)),

Dense(128, activation='relu'),

Dense(10, activation='softmax')

])

# Компилация

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

# Обучение

model.fit(X\_train, y\_train\_cat, epochs=5, batch\_size=32, validation\_split=0.1)

# Оценка

loss, acc = model.evaluate(X\_test, y\_test\_cat)

print(f"Точност на тестовия набор: {acc:.2%}")

# Визуализация на резултати

predictions = model.predict(X\_test[:10])

for i in range(10):

plt.imshow(X\_test[i], cmap='gray')

plt.title(f"Истинска: {y\_test[i]}, Прогноза: {predictions[i].argmax()}")

plt.axis('off')

plt.show()

⸻

**Какво правим:**

Използваме невронна мрежа, за да „разпознае“ коя цифра (от 0 до 9) е изписана на дадено изображение. Изображенията са черно-бели и с размер 28x28 пиксела.

**Какво включва решението:**

1. **Зареждаме данните** от keras.datasets. Получаваме тренировъчен и тестови набор.

2. **Нормализираме пикселите** – вместо стойности от 0 до 255, ги преобразуваме до стойности между 0 и 1. Това помага на модела да учи по-добре.

3. **Категоризираме отговорите** – вместо числото “5”, например, използваме [0, 0, 0, 0, 0, 1, 0, 0, 0, 0].

4. **Създаваме модел** – невронна мрежа с един скрит слой от 128 неврона и изходен слой с 10 неврона (по един за всяка цифра).

5. **Обучаваме модела** – той „гледа“ изображенията и се учи коя цифра е нарисувана.

6. **Оценяваме точността** – проверяваме колко от цифрите познава правилно върху тестовия набор.

**Защо е полезно:**

Това е класически проблем по класификация, който помага да разбереш как работят невронните мрежи за изображения и е основа на технологии като разпознаване на ръкопис или автоматизирано сканиране на документи.  
  
  
  
**Задача 2: Прогнозиране на цена на къща с невронна мрежа**

import numpy as np

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

# Данни

X = np.array([

[50, 2, 10],

[60, 3, 5],

[80, 3, 2],

[100, 4, 0],

[120, 5, 1],

[150, 6, 3]

])

y = np.array([100000, 120000, 150000, 180000, 210000, 250000])

# Модел

model = Sequential([

Dense(10, activation='relu', input\_shape=(3,)),

Dense(1)

])

# Компилация

model.compile(optimizer='adam', loss='mse')

# Обучение

model.fit(X, y, epochs=500, verbose=0)

# Прогноза

sample = np.array([[90, 4, 1]])

predicted\_price = model.predict(sample)

print(f"Прогнозирана цена за 90 кв.м, 4 стаи, 1 година: {predicted\_price[0][0]:.2f} BGN")

**Какво правим:**

Използваме невронна мрежа, за да прогнозираме **цената на къща**, базирайки се на квадратура, брой стаи и години на имота.

**Какво включва решението:**

1. **Създаваме входни данни** – всяка къща е описана с 3 характеристики.

2. **Целевата променлива** (това, което искаме да предвидим) е цената на къщата.

3. **Създаваме модел с 1 скрит слой**, който учи връзката между входовете и цената.

4. **Обучаваме модела** – той се учи от данните.

5. **Прогнозираме нова стойност** – за къща с 90 кв.м, 4 стаи и 1 година.

6. **Получаваме реалистична прогноза** – например: 195,000 BGN.

**Защо е полезно:**

Това е пример за **регресия** – т.е. предсказване на реална стойност, не категория. Използва се в имотния пазар, финансови прогнози, дори в оценка на риска.

**Задача 3: Класификация – Ще напусне ли служителят?**

import numpy as np

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from sklearn.metrics import roc\_curve, auc

import matplotlib.pyplot as plt

# Данни

X = np.array([

[25, 1, 3000],

[30, 3, 4500],

[45, 10, 7000],

[23, 0.5, 2500],

[35, 5, 6000],

[40, 8, 6500]

])

y = np.array([1, 0, 0, 1, 0, 0])

# Модел

model = Sequential([

Dense(10, activation='relu', input\_shape=(3,)),

Dense(1, activation='sigmoid')

])

# Компилация

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

# Обучение

model.fit(X, y, epochs=300, verbose=0)

# Прогноза за служител

sample = np.array([[28, 2, 4000]])

prediction = model.predict(sample)

print(f"Вероятност служителят да напусне: {prediction[0][0]:.2%}")

# ROC крива

y\_scores = model.predict(X).ravel()

fpr, tpr, thresholds = roc\_curve(y, y\_scores)

roc\_auc = auc(fpr, tpr)

plt.plot(fpr, tpr, label=f"AUC = {roc\_auc:.2f}")

plt.plot([0, 1], [0, 1], linestyle='--')

plt.xlabel("False Positive Rate")

plt.ylabel("True Positive Rate")

plt.title("ROC крива за модела")

plt.legend()

plt.show()

**Какво правим:**

Създаваме невронна мрежа, която предсказва дали даден служител ще **напусне работа** или не, базирано на възраст, години в компанията и заплата.

**Какво включва решението:**

1. **Създаваме входни данни** – служителите са описани с 3 характеристики.

2. **Целевата стойност** е 1 (напуска) или 0 (остава).

3. **Създаваме модел** с малка невронна мрежа за бинарна класификация.

4. **Обучаваме модела** и правим прогноза за нов служител.

5. **Получаваме вероятност** – напр. “С 28 години и 2 години опит вероятността да напусне е 64%”.

6. **Начертаваме ROC крива** – графика, която показва колко добре се справя моделът с класификацията.

**Защо е полезно:**

Тази задача показва как можем да използваме невронни мрежи за **HR анализи, предсказване на текучество, и вземане на управленски решения**.