## **Державний університет інформаційно-комунікаційних технологій**

**Практична робота №6**

**з дисципліни: Технології штучного інтелекту**

**Тема: Побудова та аналіз базових алгоритмів Data Science в задачах класифікації машинного навчання Machine Learning**

Виконав:  
студент ДУІКТ  
Тертишний В.Ю.

група: ШІДМ-51

м.Київ

**Мета:** Придбати практичні навички формування дата-сетів для аналізу даних Data Science в прикладних задачах класифікації та кластеризації машинного навчання Machine Learning.

**Код:**

import os

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

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

from sklearn.preprocessing import StandardScaler

from sklearn.decomposition import PCA

from sklearn.metrics import accuracy\_score, confusion\_matrix, classification\_report

from sklearn.linear\_model import LogisticRegression

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

import tensorflow as tf

# Шлях до файлу

file\_path = '**>ШЛЯХ ДО ФАЙЛУ ПР6 СЮДИ<**'

# Перевіряємо, чи файл існує

if os.path.exists(file\_path):

print("Файл знайдений!")

# Завантажуємо дані

df = pd.read\_csv(file\_path)

else:

print("Файл не знайдений. Перевірте шлях.")

# Завершуємо виконання коду, якщо файл не знайдено

exit()

# Перевіримо структуру даних

print(df.info())

print(df.describe())

# Нормалізація даних

features = df.drop('DEATH\_EVENT', axis=1)

target = df['DEATH\_EVENT']

scaler = StandardScaler()

features\_scaled = scaler.fit\_transform(features)

# Застосуємо метод головних компонент (PCA) для кластеризації

pca = PCA(n\_components=2) # Зведемо до двох вимірів для візуалізації

features\_pca = pca.fit\_transform(features\_scaled)

df\_pca = pd.DataFrame(features\_pca, columns=['PC1', 'PC2'])

df\_pca['DEATH\_EVENT'] = target

# Візуалізація кластерів

plt.figure(figsize=(8, 6))

sns.scatterplot(x='PC1', y='PC2', hue='DEATH\_EVENT', data=df\_pca, palette="viridis")

plt.title("Кластеризація пацієнтів за PCA")

plt.xlabel("PC1")

plt.ylabel("PC2")

plt.show()

# Створимо нейронну мережу для класифікації

model = Sequential()

model.add(Dense(64, input\_dim=features\_scaled.shape[1], activation='relu'))

model.add(Dense(32, activation='relu'))

model.add(Dense(1, activation='sigmoid'))

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

# Розділимо дані на тренувальний і тестовий набори

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

# Тренування нейронної мережі

history = model.fit(X\_train, y\_train, epochs=50, batch\_size=10, validation\_data=(X\_test, y\_test))

# Оцінка точності на тестовій вибірці

y\_pred\_nn = (model.predict(X\_test) > 0.5).astype("int32")

print("Нейронна мережа - Оцінка точності:")

print(accuracy\_score(y\_test, y\_pred\_nn))

print(confusion\_matrix(y\_test, y\_pred\_nn))

print(classification\_report(y\_test, y\_pred\_nn))

# Логістична регресія для порівняння

log\_reg = LogisticRegression()

log\_reg.fit(X\_train, y\_train)

y\_pred\_lr = log\_reg.predict(X\_test)

print("Логістична регресія - Оцінка точності:")

print(accuracy\_score(y\_test, y\_pred\_lr))

print(confusion\_matrix(y\_test, y\_pred\_lr))

print(classification\_report(y\_test, y\_pred\_lr))

# Візуалізація точності моделі логістичної регресії

plt.figure(figsize=(8, 6))

sns.heatmap(confusion\_matrix(y\_test, y\_pred\_lr), annot=True, fmt="d", cmap="Blues")

plt.title("Матриця невідповідностей для логістичної регресії")

plt.xlabel("Прогнозоване значення")

plt.ylabel("Справжнє значення")

plt.show()

# Розрахунок кореляційної матриці

corr\_matrix = df.corr()

plt.figure(figsize=(12, 10))

sns.heatmap(corr\_matrix, annot=True, cmap="coolwarm")

plt.title("Кореляційна матриця змінних")

plt.show()

# Зберігаємо результати в файл

with open("heart\_failure\_analysis\_results.txt", "w") as file:

file.write("Оцінка моделі нейронної мережі:\n")

file.write(classification\_report(y\_test, y\_pred\_nn))

file.write("\nОцінка логістичної регресії:\n")

file.write(classification\_report(y\_test, y\_pred\_lr))

print("Аналіз завершено. Результати збережено у файлі 'heart\_failure\_analysis\_results.txt'.")

**Робота коду:**  
(**>Скріни сюди<**)

**Висновки**

У ході виконання поставленої мети ми ознайомилися з основами проектування чат-ботів за допомогою Python та бібліотеки TeleBot, навчилися структурувати код, створювати логіку бота та забезпечувати його функціональність. Було засвоєно принципи взаємодії чат-ботів із користувачами, включаючи обробку текстових команд і повідомлень. Ми створили функціональні клавіатури (як стандартні, так і онлайн-клавіатури) для полегшення роботи користувача з ботом. Також було реалізовано можливості роботи чат-ботів із мультимедійними даними, такими як зображення та файли, що розширило функціональність ботів і підвищило зручність для користувачів. Таким чином, виконання поставленої мети дало змогу отримати теоретичні знання та практичні навички створення багатофункціональних чат-ботів із використанням сучасних технологій і бібліотек.