МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Тернопільський національний технічний університет
імені Івана Пулюя

Кафедра програмної інженерії

Методичні вказівки до практичних робіт

з дисципліни

«Ідентифікація складних систем і об’єктів»

CEH ID: **6687**

для здобувачів третього (освітньо-наукового) рівня вищої освіти ОНП
«Інженерія програмного забезпечення»

Тернопіль — 2025

УДК 004.4(075.8)
ББК 32.973.202я73
Б83

Укладач:
В.М. Бревус, PhD

Рецензент:
В.В. Яцишин, к.т.н., доцент
завідувач кафедри систем штучного інтелекту та аналізу даних ТНТУ ім. І. Пулюя

Розглянуто й затверджено на засіданні кафедри програмної інженерії.
Протокол № 1 від 01.09.2025.

Розглянуто й затверджено на засіданні методичної комісії факультету комп'ютерно-інформаційних систем та програмної інженерії Тернопільського національного технічного університету імені Івана Пулюя.
Протокол № 1 від 01.09.2025.

Б83 Методичні вказівки до практичних робіт з дисципліни «Ідентифікація складних систем і об'єктів» для здобувачів третього (освітньо-наукового) рівня вищої освіти ОНП «Інженерія програмного забезпечення» / Укладач: Бревус В.М. – Тернопіль: ТНТУ імені Івана Пулюя, 2025 – 32 с.

Відповідальний за випуск: М.Р. Петрик, докт. фіз.-мат. наук, професор

АНОТАЦІЯ

У збірнику практичних робіт представлено комплексний наскрізний проєкт «Система ідентифікації та класифікації біомедичних сигналів», присвячений розробці системи діагностики ранніх ознак хвороби Паркінсона на основі аналізу спіральних малюнків.

Практичні роботи охоплюють весь цикл розробки системи машинного навчання: від підготовки та аналізу біомедичних даних до валідації та розгортання готової системи. Особливу увагу приділено застосуванню методів ідентифікації систем у медичних додатках, включаючи непараметричні та параметричні підходи, спектральний аналіз, інженерію ознак та етичні аспекти використання штучного інтелекту в медицині.

Збірник практичних робіт призначений для здобувачів третього (освітньо-наукового) рівня вищої освіти, які навчаються за освітньо-науковою програмою «Інженерія програмного забезпечення». Він також буде корисним для аспірантів, науковців та інженерів, що спеціалізуються в галузі біомедичної інженерії та машинного навчання.

Ключові слова: ідентифікація систем, біомедичні сигнали, машинне навчання, хвороба Паркінсона, спектральний аналіз, класифікація.

ABSTRACT

The collection of practical works presents a comprehensive end-to-end project “System for Identification and Classification of Biomedical Signals” dedicated to developing a diagnostic system for early signs of Parkinson’s disease based on spiral drawing analysis.

The practical works cover the complete machine learning system development cycle: from biomedical data preparation and analysis to validation and deployment of the finished system. Special attention is given to the application of system identification methods in medical applications, including non-parametric and parametric approaches, spectral analysis, feature engineering, and ethical aspects of artificial intelligence use in medicine.

The collection of practical works is intended for students of the third (educational and scientific) level of higher education enrolled in the “Software Engineering” educational and scientific program. It will also be useful for postgraduate students, researchers, and engineers specializing in biomedical engineering and machine learning.

Keywords: system identification, biomedical signals, machine learning, Parkinson’s disease, spectral analysis, classification.

Зміст

Вступ

Деталі щодо оформлення програмного коду та змісту звіту практичних робіт узгоджуються з викладачем у системі електронного навчання відповідного курсу. Приклад оформлення звіту з титульним аркушем, структурою та рекомендаціями наведено в додатку А.

Усі практичні роботи повинні бути виконані з дотриманням принципів академічної доброчесності, відповідно до Положення про академічну доброчесність учасників освітнього процесу Тернопільського національного технічного університету імені Івана Пулюя.

Наскрізний проєкт

«Система ідентифікації та класифікації біомедичних сигналів: від збору даних до машинного навчання»

Протягом всіх практичних робіт здобувачі розробляють комплексну систему для діагностики ранніх ознак хвороби Паркінсона на основі аналізу спіральних малюнків, зібраних за допомогою цифрових планшетів. Проєкт базується на дослідженні *«Digitized spiral drawing classification for Parkinson's disease diagnosis»* [1], яке продемонструвало 91% точність класифікації.

Медичне обґрунтування

Хвороба Паркінсона — найпоширеніше нейродегенеративне захворювання, що вражає понад 10 мільйонів людей у світі. Раннє виявлення критично важливе для ефективного лікування, однак сучасні методи діагностики часто виявляють захворювання лише на пізніх стадіях. Дослідження [2] підтвердило ефективність методів аналізу темпоральних нерегулярностей у спіральних малюнках для оцінки симптомів хвороби Паркінсона з високою надійністю тест-ретест.

Ключові переваги підходу:

- **Неінвазивність:** Тестування не потребує медичних процедур
- **Доступність:** Використання стандартних цифрових планшетів
- **Об'єктивність:** Кількісні метрики замість суб'єктивної оцінки
- **Чутливість:** Виявлення тонких моторних порушень
- **Масштабованість:** Можливість використання в телемедицині

Технічний підхід

Проєкт інтегрує сучасні методи ідентифікації систем з передовими технологіями машинного навчання:

1. **Сигнальна обробка:** Фільтрація, згладжування, обчислення похідних

2. **Непараметрична ідентифікація:** Кореляційний та спектральний аналіз [2]
3. **Параметрична ідентифікація:** Класифікація з використанням ML
4. **Валідація та розгортання:** Клінічна оцінка та практичне впровадження

Структура практичних робіт

ПР№1: Аналіз та підготовка біомедичних даних

- Завантаження та preprocessing датасету спіральних малюнків [3]
- Обчислення кінематичних параметрів (швидкість, прискорення, кривизна) [4]
- Експлоративний аналіз даних та виявлення початкових закономірностей [2]
- Створення інфраструктури для подальшої обробки

ПР№2: Непараметрична ідентифікація біомедичних систем

- Кореляційний аналіз траєкторій малювання [5]
- Спектральний аналіз для виявлення тремору (4-12 Гц) [6]
- Аналіз фазових портретів та динамічних характеристик [2]
- Видобування ознак у частотній області [1]

ПР№3: Параметрична ідентифікація та машинне навчання

- Комплексна інженерія ознак на основі попередніх робіт [5]
- Порівняння алгоритмів класифікації (SVM, Random Forest, Neural Networks) [6]
- Оптимізація гіперпараметрів та відбір ознак [3]
- Валідація моделей з медично релевантними метриками [4]

ПР№4: Валідація та розгортання системи

- Комплексна валідація на зовнішніх датасетах [7]
- Аналіз справедливості алгоритмів та етичних аспектів [5]
- Створення інтерпретованої системи з поясненнями [2]
- Підготовка до клінічного впровадження [7]

Очікувані результати

По завершенні проєкту здобувачі матимуть:

1. **Функціональну систему діагностики** з точністю 85-95%
2. **Глибоке розуміння** методів ідентифікації біомедичних систем
3. **Практичні навички** роботи з реальними медичними даними
4. **Досвід валідації** систем штучного інтелекту для медицини
5. **Портфоліо проєкту** для професійного розвитку

Зв'язок з теоретичним курсом

Практичні роботи тісно інтегровані з лекційним матеріалом:

- **Лекції 1-2:** Основи ідентифікації — ПР№1 (підготовка даних)
- **Лекції 3-4:** Непараметричні методи — ПР№2 (спектральний аналіз)
- **Лекція 5:** Параметричні методи — ПР№3 (машинне навчання)
- **Лекції 6-7:** Нелінійні системи та валідація — ПР№4 (розгортання)

Інструментарій та ресурси

Програмне забезпечення: Python 3.8+ з бібліотеками scikit-learn, scipy, pandas, matplotlib, seaborn

Дані: Основний датасет UCI Machine Learning Repository [8] містить спіральні малюнки, зібрані за допомогою цифрового графічного планшета від пацієнтів з хворобою Паркінсона та здорових контрольних суб'єктів

Обчислювальні ресурси: Стандартний ноутбук або комп'ютер, можливість використання хмарних платформ

Практична робота №1

Аналіз та підготовка біомедичних даних для ідентифікації систем

Мета роботи

Ознайомитися з основними принципами збору, обробки та аналізу біомедичних сигналів на прикладі спіральних малюнків для діагностики хвороби Паркінсона. Освоїти методи попередньої обробки даних та їх візуалізації.

Наскрізний проєкт

«Система ідентифікації та класифікації біомедичних сигналів»

Протягом всіх практичних робіт ми будемо розробляти комплексну систему для аналізу спіральних малюнків, зібраних за допомогою цифрових планшетів, з метою виявлення ранніх ознак хвороби Паркінсона. Проєкт базується на дослідженні *«Digitized spiral drawing classification for Parkinson's disease diagnosis»* [1].

Теоретичні відомості

Хвороба Паркінсона — найпоширеніше нейродегенеративне захворювання, що значно впливає на моторні функції літніх людей. Ранніми симптомами є порушення почерку та тремор при виконанні повторюваних завдань, зокрема малювання спіралей.

Цифрові планшети дозволяють збирати детальні кінематичні дані про процес малювання, включаючи:

- Координати точок (x, y) в часі
- Швидкість переміщення пера
- Прискорення
- Тиск на поверхню
- Рухи «в повітрі» (без контакту з поверхнею)

У дослідженні використовувався датасет з 25 пацієнтів з хворобою Паркінсона та 15 здорових осіб контрольної групи.

Завдання

Завдання 1.1. Завантаження та аналіз датасету

1. Завантажити датасет спіральних малюнків з UCI Machine Learning Repository [8]:
    - Використати відкриті дані з 77 суб'єктів (62 пацієнти + 15 контроль)

- Проаналізувати структуру файлів з координатами та часовими мітками

2. Проаналізувати структуру даних:

   - Формат зберігання координат та часових міток
   - Параметри дискретизації
   - Метадані про пацієнтів

Завдання 1.2. Створення інфраструктури для обробки даних Розробити Python-модуль для роботи з даними:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy import signal
import seaborn as sns

class SpiralDataProcessor:
    def __init__(self):
        self.data = None
        self.metadata = None

    def load_data(self, filepath):
        """Завантаження даних спірального малюнка"""
        # Реалізація завантаження
        pass

    def preprocess(self):
        """Попередня обробка даних"""
        # Фільтрація шуму
        # Нормалізація координат
        # Обчислення похідних
        pass

    def visualize_trajectory(self, patient_id):
        """Візуалізація траєкторії малюнка"""
        plt.figure(figsize=(10, 8))
        # Код візуалізації
        pass
```

Завдання 1.3. Базовий аналіз кінематичних параметрів

1. Обчислити похідні характеристики:

$$v(t) = \sqrt{\left(\frac{dx}{dt}\right)^2 + \left(\frac{dy}{dt}\right)^2} \tag{1}$$

$$a(t) = \frac{dv}{dt} \tag{2}$$

$$\kappa(t) = \frac{x'y'' - y'x''}{(x'^2 + y'^2)^{3/2}} \tag{3}$$

2. Статистичні показники:

   - Середня та максимальна швидкість
   - Варіація швидкості (CV = σ_v/μ_v)
   - Середнє та максимальне прискорення
   - Загальна тривалість малювання
   - Кількість зупинок (швидкість $< threshold$)

Завдання 1.4. Експлоративний аналіз даних

1. Створити порівняльні візуалізації:

   - Накладення траєкторій здорових vs хворих
   - Гістограми розподілу швидкостей
   - Box plots для ключових параметрів

2. Провести статистичні тести:

   - t-тест для порівняння середніх значень
   - Тест Манна-Уітні для непараметричного порівняння
   - Аналіз кореляції між параметрами

Інструментарій

Python 3.8+ з бібліотеками:

- **pandas, numpy** — обробка даних
- **matplotlib, seaborn** — візуалізація
- **scipy** — сигнальна обробка та статистика
- **scikit-learn** — машинне навчання (для майбутніх робіт)

Приклад оформлення коду проєкту доступний у файлі: `Workshop/src/SE-tremor-identification/Practice-1-Biomedical-Data-Analysis.ipynb` репозиторії курсу: https://github.com/TNTU-F2-SE-PhD/SE-PhD-Identification-of-complex-systems-and-objects

Очікувані результати

1. Модуль для завантаження та обробки даних спіральних малюнків
2. Набір функцій для обчислення кінематичних параметрів
3. Візуалізації, що демонструють відмінності між групами
4. Звіт з первинним аналізом характеристик датасету
5. Рекомендації щодо найбільш інформативних параметрів

Контрольні питання

1. Які фізіологічні механізми призводять до змін у моториці при хворобі Паркінсона?
2. Чому спіральні малюнки є більш чутливими до моторних порушень, ніж прямі лінії?
3. Як частота дискретизації впливає на точність обчислення похідних?
4. Які методи фільтрації доцільно застосовувати до біомедичних сигналів?
5. Як забезпечити відтворюваність результатів при роботі з реальними даними?

Практична робота №2

Непараметрична ідентифікація біомедичних систем

Мета роботи

Застосувати методи непараметричної ідентифікації для виявлення характеристик біомедичних сигналів без припущень про структуру моделі. Освоїти методи кореляційного та спектрального аналізу для характеристики динамічних властивостей рухів при малюванні спіралей.

Продовження наскрізного проєкту

У цій роботі ми розвиваємо систему аналізу, створену в ПР№1, додаючи методи непараметричної ідентифікації для глибшого розуміння динамічних властивостей моторної активності при хворобі Паркінсона.

Теоретичні відомості

Непараметрична ідентифікація — це підхід до аналізу систем, що не вимагає попередніх припущень про математичну структуру моделі. Основні методи:

1. **Кореляційний аналіз** — вивчення статистічних зв'язків між сигналами
2. **Спектральний аналіз** — дослідження частотних характеристик
3. **Аналіз імпульсної характеристики** — реакція системи на одиничний імпульс
4. **Аналіз перехідної характеристики** — реакція на ступінчасту зміну входу

У контексті біомедичних сигналів ці методи дозволяють виявити:

- Частотні компоненти тремору
- Затримки в нейромоторній системі
- Властивості автокореляції рухів
- Спектральну потужність у різних діапазонах

Завдання

Завдання 2.1. Кореляційний аналіз траєкторій

1. Обчислити автокореляційні функції для координат x(t) та y(t):

$$R_{xx}(\tau) = \mathbb{E}[x(t) \cdot x(t+\tau)]$$

2. Обчислити взаємокореляцію між x(t) та y(t):

$$R_{xy}(\tau) = \mathbb{E}[x(t) \cdot y(t+\tau)]$$

3. Проаналізувати часові затримки та періодичності в рухах

Завдання 2.2. Спектральний аналіз Реалізувати спектральний аналіз кінематичних параметрів:

```
from scipy.fft import fft, fftfreq
from scipy.signal import welch, spectrogram

def spectral_analysis(signal, fs=100):
    """
    Виконує спектральний аналіз сигналу

    Args:
        signal: одновимірний сигнал
        fs: частота дискретизації

    Returns:
        frequencies, power_spectrum
    """
    # FFT аналіз
    N = len(signal)
    yf = fft(signal)
    xf = fftfreq(N, 1/fs)

    # Welch метод для оцінки спектральної щільності
    f_welch, Pxx = welch(signal, fs, nperseg=min(256, N//4))

    return xf[:N//2], np.abs(yf[:N//2]), f_welch, Pxx

def tremor_frequency_analysis(velocity_signal, fs=100):
    """
    Аналіз частот тремору (4-12 Hz для Паркінсона)
    """
    freqs, psd = welch(velocity_signal, fs, nperseg=256)

    # Діапазони частот
    tremor_band = (freqs >= 4) & (freqs <= 12)
    total_power = np.trapz(psd, freqs)
```

```
    tremor_power = np.trapz(psd[tremor_band], freqs[tremor_band])

    return tremor_power / total_power
```

Завдання 2.3. Аналіз динамічних характеристик

1. Оцінити імпульсну характеристику системи через деконволюцію
2. Проаналізувати перехідні процеси в швидкості
3. Виміряти характерні часи системи (час встановлення, час наростання)

Завдання 2.4. Дослідження нелінійностей

1. Побудувати фазові портрети (x vs. dx/dt, y vs. dy/dt)
2. Проаналізувати гістерезис у рухах
3. Обчислити показники хаотичності (якщо застосовно)

Реалізація ключових алгоритмів

Модуль для непараметричного аналізу:

```
class NonparametricAnalyzer:
    def __init__(self, data):
        self.data = data
        self.fs = 100  # Частота дискретизації

    def compute_autocorrelation(self, signal, max_lag=None):
        """Обчислення автокореляції"""
        from scipy.signal import correlate

        if max_lag is None:
            max_lag = len(signal) // 4

        correlation = correlate(signal, signal, mode='full')
        lags = np.arange(-len(signal)+1, len(signal))

        # Вибрати тільки позитивні лаги
        positive_lags = lags[len(lags)//2:][:max_lag]
        positive_corr = correlation[len(correlation)//2:][:max_lag]

        # Нормалізація
        positive_corr = positive_corr / positive_corr[0]
```

```
        return positive_lags / self.fs, positive_corr

    def frequency_domain_features(self, signal):
        """Видобування ознак у частотній області"""
        freqs, psd = welch(signal, self.fs, nperseg=256)

        features = {
            'dominant_freq': freqs[np.argmax(psd)],
            'spectral_centroid': np.sum(freqs * psd) / np.sum(psd),
            'spectral_rolloff': self.spectral_rolloff(freqs, psd, 0.85),
            'tremor_ratio': self.tremor_power_ratio(freqs, psd)
        }

        return features

    def spectral_rolloff(self, freqs, psd, threshold=0.85):
        """Частота, нижче якої зосереджено threshold% енергії"""
        cumulative_power = np.cumsum(psd)
        total_power = cumulative_power[-1]
        rolloff_index = np.where(cumulative_power >= threshold * total_power)[0][0]
        return freqs[rolloff_index]

    def tremor_power_ratio(self, freqs, psd):
        """Відношення потужності в діапазоні тремору до загальної"""
        tremor_mask = (freqs >= 4) & (freqs <= 12)
        tremor_power = np.trapz(psd[tremor_mask], freqs[tremor_mask])
        total_power = np.trapz(psd, freqs)
        return tremor_power / total_power if total_power > 0 else 0
```

Аналіз результатів

1. **Порівняльний аналіз груп:**
   - Створити таблиці спектральних характеристик для здорових та хворих
   - Побудувати візуалізації розподілів ключових параметрів
   - Проаналізувати статистичну значущість відмінностей
2. **Виявлення біомаркерів:**
   - Визначити найбільш дискримінативні частотні діапазони
   - Оцінити кореляції між різними параметрами

- Ранжувати параметри за інформативністю

Очікувані результати

1. Модуль непараметричного аналізу з реалізованими методами
2. Комплексний аналіз спектральних характеристик датасету
3. Візуалізації автокореляційних та спектральних функцій
4. Таблиці порівняння груп за непараметричними показниками
5. Рекомендації щодо найбільш інформативних характеристик

Контрольні питання

1. У чому переваги непараметричних методів для біомедичних сигналів?
2. Як інтерпретувати піки в спектрі швидкості руху?
3. Чому тремор при хворобі Паркінсона проявляється в діапазоні 4-12 Гц?
4. Як автокореляційна функція може виявити періодичні компоненти?
5. Які обмеження мають непараметричні методи порівняно з параметричними?

Практична робота №3

Параметрична ідентифікація та машинне навчання для класифікації

Мета роботи

Розробити систему класифікації для розпізнавання ранніх ознак хвороби Паркінсона на основі параметричних моделей та алгоритмів машинного навчання. Порівняти ефективність різних підходів до класифікації біомедичних сигналів.

Розвиток наскрізного проєкту

У цій роботі ми інтегруємо результати попередніх робіт для створення повноцінної системи класифікації. Використаємо ознаки, отримані в ПР№1-2, для побудови моделей машинного навчання та їх оптимізації.

Теоретичні відомості

Параметрична ідентифікація в контексті машинного навчання передбачає створення моделей з конкретною структурою та оцінкою параметрів на основі даних. Для біомедичних сигналів найбільш ефективними є:

1. **Лінійні моделі:** Логістична регресія, SVM з лінійним ядром
2. **Ансамблі моделей:** Random Forest, Gradient Boosting
3. **Нейронні мережі:** MLP, CNN для послідовностей

Ключові етапи:

- Інженерія ознак (feature engineering)
- Відбір ознак (feature selection)
- Навчання та валідація моделей
- Оптимізація гіперпараметрів
- Оцінка якості класифікації

Завдання

Завдання 3.1. Інженерія ознак Створити комплексний набір ознак на основі результатів попередніх робіт:

```
class FeatureExtractor:
    def __init__(self):
        self.features = {}

    def extract_temporal_features(self, trajectory_data):
        """Часові характеристики"""
        features = {
            'total_time': trajectory_data['time'].max(),
            'mean_velocity': np.mean(trajectory_data['velocity']),
            'std_velocity': np.std(trajectory_data['velocity']),
            'cv_velocity': np.std(trajectory_data['velocity']) /
                           np.mean(trajectory_data['velocity']),
            'max_acceleration': np.max(trajectory_data['acceleration']),
            'num_velocity_peaks': len(find_peaks(trajectory_data['velocity'])[0]),
            'mean_jerk': np.mean(np.abs(np.diff(trajectory_data['acceleration'])))
        }
        return features

    def extract_spatial_features(self, x, y):
        """Просторові характеристики"""
        # Обчислення площі та периметру
        area = np.trapz(y, x)  # Приблизна площа
        perimeter = np.sum(np.sqrt(np.diff(x)**2 + np.diff(y)**2))

        # Центр мас та дисперсія
        centroid_x = np.mean(x)
        centroid_y = np.mean(y)
        dispersion = np.mean((x - centroid_x)**2 + (y - centroid_y)**2)

        return {
            'spiral_area': abs(area),
            'spiral_perimeter': perimeter,
            'compactness': 4 * np.pi * abs(area) / (perimeter**2),
            'centroid_x': centroid_x,
            'centroid_y': centroid_y,
            'spatial_dispersion': dispersion
        }

    def extract_frequency_features(self, signal, fs=100):
        """Частотні характеристики з ПР№2"""
```

```
        freqs, psd = welch(signal, fs, nperseg=256)

        # Статистики спектру
        spectral_centroid = np.sum(freqs * psd) / np.sum(psd)
        spectral_rolloff = self.compute_rolloff(freqs, psd, 0.85)
        spectral_bandwidth = np.sqrt(np.sum(((freqs - spectral_centroid)**2) * psd) / np

        # Тремор-специфічні ознаки
        tremor_mask = (freqs >= 4) & (freqs <= 12)
        tremor_power = np.trapz(psd[tremor_mask], freqs[tremor_mask])
        total_power = np.trapz(psd, freqs)

        return {
            'spectral_centroid': spectral_centroid,
            'spectral_rolloff': spectral_rolloff,
            'spectral_bandwidth': spectral_bandwidth,
            'tremor_power_ratio': tremor_power / total_power,
            'dominant_frequency': freqs[np.argmax(psd)]
        }
```

Завдання 3.2. Побудова та порівняння моделей Реалізувати кілька алгоритмів класифікації:

```
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import cross_val_score, GridSearchCV
from sklearn.metrics import classification_report, confusion_matrix

class ParkinsonClassifier:
    def __init__(self):
        self.models = {
            'logistic': LogisticRegression(random_state=42),
            'svm': SVC(random_state=42, probability=True),
            'random_forest': RandomForestClassifier(random_state=42),
            'gradient_boosting': GradientBoostingClassifier(random_state=42),
            'neural_network': MLPClassifier(random_state=42, max_iter=1000)
        }
        self.results = {}

    def train_models(self, X_train, y_train):
```

```
    """Навчання всіх моделей"""
    for name, model in self.models.items():
        print(f"Training {name}...")
        model.fit(X_train, y_train)

        # Cross-validation
        cv_scores = cross_val_score(model, X_train, y_train, cv=5)
        self.results[name] = {
            'cv_mean': cv_scores.mean(),
            'cv_std': cv_scores.std(),
            'model': model
        }

def optimize_hyperparameters(self, X_train, y_train):
    """Оптимізація гіперпараметрів для найкращої моделі"""
    param_grids = {
        'random_forest': {
            'n_estimators': [100, 200, 300],
            'max_depth': [10, 20, None],
            'min_samples_split': [2, 5, 10]
        },
        'svm': {
            'C': [0.1, 1, 10, 100],
            'gamma': ['scale', 'auto', 0.001, 0.01],
            'kernel': ['rbf', 'linear']
        }
    }

    best_models = {}
    for model_name, param_grid in param_grids.items():
        grid_search = GridSearchCV(
            self.models[model_name],
            param_grid,
            cv=5,
            scoring='accuracy',
            n_jobs=-1
        )
        grid_search.fit(X_train, y_train)
        best_models[model_name] = grid_search.best_estimator_
```

```
    return best_models
```

Завдання 3.3. Відбір ознак Реалізувати методи відбору найбільш інформативних ознак:

1. **Статистичні методи:** ANOVA F-test, взаємна інформація
2. **Рекурсивний відбір:** RFE (Recursive Feature Elimination)
3. **Regularization-based:** LASSO для автоматичного відбору
4. **Tree-based:** важливість ознак з Random Forest

Завдання 3.4. Валідація та тестування

1. Розділити дані на тренувальну/валідаційну/тестову вибірки (60%/20%/20%)
2. Провести k-fold cross-validation
3. Обчислити метрики якості: accuracy, precision, recall, F1-score, AUC-ROC
4. Побудувати confusion matrix та ROC криві
5. Проаналізувати помилки класифікації

Метрики оцінки якості

Для біомедичної класифікації особливо важливі:

$$\text{Sensitivity (Recall)} = \frac{TP}{TP + FN} \tag{4}$$

$$\text{Specificity} = \frac{TN}{TN + FP} \tag{5}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{6}$$

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{7}$$

де TP — true positive, TN — true negative, FP — false positive, FN — false negative.

Інтерпретація результатів

```
def interpret_model_results(model, feature_names, X_test, y_test):
    """Інтерпретація результатів моделі"""

    # Важливість ознак для tree-based моделей
    if hasattr(model, 'feature_importances_'):
        importance_df = pd.DataFrame({
            'feature': feature_names,
```

```
            'importance': model.feature_importances_
        }).sort_values('importance', ascending=False)

        plt.figure(figsize=(10, 6))
        sns.barplot(data=importance_df.head(15), x='importance', y='feature')
        plt.title('Важливість ознак')
        plt.show()

# SHAP values для глибшої інтерпретації
import shap
if hasattr(model, 'predict_proba'):
    explainer = shap.Explainer(model)
    shap_values = explainer(X_test)
    shap.summary_plot(shap_values, X_test, feature_names=feature_names)
```

Очікувані результати

1. Комплексна система видобування ознак з біомедичних сигналів
2. Порівняльний аналіз ефективності різних алгоритмів класифікації
3. Оптимізовані моделі з найкращими гіперпараметрами
4. Аналіз важливості ознак та їх інтерпретація
5. Валідація моделей з метриками, релевантними для медичної діагностики
6. Рекомендації щодо найкращого підходу для практичного застосування

Контрольні питання

1. Чому важливо використовувати cross-validation у біомедичних застосуваннях?
2. Як інтерпретувати trade-off між sensitivity та specificity?
3. Які ознаки найбільш інформативні для виявлення хвороби Паркінсона?
4. Як можна підвищити надійність класифікації в умовах малих датасетів?
5. Які етичні аспекти слід враховувати при розробці медичних систем ШІ?

Практична робота №4

Валідація та розгортання системи ідентифікації

Мета роботи

Завершити розробку наскрізного проєкту створенням комплексної системи валідації, аналізу продуктивності та підготовки до практичного застосування системи діагностики хвороби Паркінсона на основі аналізу спіральних малюнків.

Завершення наскрізного проєкту

У цій фінальній роботі ми об'єднуємо всі компоненти, розроблені в попередніх практичних роботах, в єдину, готову до застосування систему. Особлива увага приділяється валідації, надійності та етичним аспектам медичної діагностики.

Теоретичні відомості

Валідація медичних систем ШІ — критично важливий процес, що включає:

1. **Клінічну валідацію** — тестування в реальних умовах
2. **Статистичну валідацію** — строгий аналіз метрик
3. **Розбіжну валідацію** — тестування на різних популяціях
4. **Темпоральну валідацію** — стабільність у часі

Ключові вимоги:

- Інтерпретованість результатів
- Надійність і відтворюваність
- Справедливість алгоритмів
- Конфіденційність даних
- Регуляторна відповідність

Завдання

Завдання 4.1. Комплексна валідація системи Провести всебічну оцінку розробленої системи:

```
class SystemValidator:
    def __init__(self, models, feature_extractor, data_processor):
        self.models = models
        self.feature_extractor = feature_extractor
```

```
        self.data_processor = data_processor
        self.validation_results = {}

    def cross_dataset_validation(self, train_data, external_data):
        """Валідація на зовнішніх датасетах"""
        results = {}

        for model_name, model in self.models.items():
            # Навчання на основному датасеті
            X_train = self.feature_extractor.extract_features(train_data)
            y_train = train_data['labels']
            model.fit(X_train, y_train)

            # Тестування на зовнішньому датасеті
            X_external = self.feature_extractor.extract_features(external_data)
            y_external = external_data['labels']
            y_pred = model.predict(X_external)

            results[model_name] = self.compute_detailed_metrics(
                y_external, y_pred, model.predict_proba(X_external)[:, 1]
            )

        return results

    def temporal_stability_test(self, data_with_timestamps):
        """Тест стабільності в часі"""
        # Розділити дані за часовими періодами
        periods = self.split_by_time(data_with_timestamps)
        stability_metrics = {}

        for period_name, period_data in periods.items():
            X = self.feature_extractor.extract_features(period_data)
            y_true = period_data['labels']

            period_results = {}
            for model_name, model in self.models.items():
                y_pred = model.predict_proba(X)[:, 1]
                period_results[model_name] = roc_auc_score(y_true, y_pred)

            stability_metrics[period_name] = period_results
```

```
        return stability_metrics

    def demographic_fairness_analysis(self, data_with_demographics):
        """Аналіз справедливості алгоритмів"""
        fairness_results = {}

        demographic_groups = ['age_group', 'gender', 'ethnicity']

        for group in demographic_groups:
            group_results = {}
            for group_value in data_with_demographics[group].unique():
                subset = data_with_demographics[
                    data_with_demographics[group] == group_value
                ]

                X = self.feature_extractor.extract_features(subset)
                y_true = subset['labels']

                for model_name, model in self.models.items():
                    y_pred_proba = model.predict_proba(X)[:, 1]
                    auc = roc_auc_score(y_true, y_pred_proba)
                    group_results[f"{group_value}_{model_name}"] = auc

            fairness_results[group] = group_results

        return fairness_results
```

Завдання 4.2. Аналіз помилок та покращення Провести детальний аналіз помилок системи:

1. **Аналіз false positives:** Які характеристики здорових людей призводять до помилкової класифікації?
2. **Аналіз false negatives:** Які типи хвороби Паркінсона найскладніше виявити?
3. **Edge cases:** Ідентифікація граничних випадків та їх обробка
4. **Стратегії покращення:** Збір додаткових даних, аугментація, ансамблі

Завдання 4.3. Створення інтерпретованої системи Розробити модуль для пояснення рішень системи:

```
class ExplainableAI:
    def __init__(self, model, feature_names):
        self.model = model
        self.feature_names = feature_names

    def generate_patient_report(self, patient_features, patient_id):
        """Генерація персоналізованого звіту"""
        prediction = self.model.predict_proba([patient_features])[0]
        risk_score = prediction[1]  # Ймовірність хвороби Паркінсона

        # SHAP пояснення
        import shap
        explainer = shap.Explainer(self.model)
        shap_values = explainer([patient_features])

        # Топ-5 найвпливовіших ознак
        feature_importance = np.abs(shap_values.values[0])
        top_features_idx = np.argsort(feature_importance)[-5:]

        report = {
            'patient_id': patient_id,
            'risk_score': risk_score,
            'risk_category': self.categorize_risk(risk_score),
            'key_indicators': [
                {
                    'feature': self.feature_names[idx],
                    'value': patient_features[idx],
                    'impact': shap_values.values[0][idx],
                    'interpretation': self.interpret_feature(
                        self.feature_names[idx],
                        patient_features[idx],
                        shap_values.values[0][idx]
                    )
                } for idx in top_features_idx
            ],
            'recommendations': self.generate_recommendations(
                patient_features, risk_score
            )
        }
```

```
        return report

    def categorize_risk(self, risk_score):
        """Категоризація ризику"""
        if risk_score < 0.3:
            return "Низький ризик"
        elif risk_score < 0.7:
            return "Помірний ризик"
        else:
            return "Високий ризик"

    def interpret_feature(self, feature_name, value, impact):
        """Інтерпретація впливу ознаки"""
        interpretations = {
            'tremor_power_ratio': {
                'high_positive': "Підвищена активність в діапазоні тремору",
                'low_negative': "Нормальна активність в діапазоні тремору"
            },
            'cv_velocity': {
                'high_positive': "Значна варіабельність швидкості руху",
                'low_negative': "Стабільна швидкість руху"
            }
            # Додати інтерпретації для всіх ознак
        }

        if feature_name in interpretations:
            if impact > 0:
                return interpretations[feature_name].get(
                    'high_positive',
                    f"Підвищене значення ({value:.3f}) збільшує ризик"
                )
            else:
                return interpretations[feature_name].get(
                    'low_negative',
                    f"Знижене значення ({value:.3f}) зменшує ризик"
                )

        return f"Значення {value:.3f} {'збільшує' if impact > 0 else 'зменшує'} ризик"
```

Завдання 4.4. Підготовка до розгортання Створити систему готову до практичного застосування:

1. **API інтерфейс:** REST API для інтеграції з медичними системами
2. **Користувацький інтерфейс:** Веб-додаток для лікарів
3. **Документація:** Технічна та медична документація
4. **Тестування безпеки:** Захист персональних даних
5. **Моніторинг продуктивності:** Система відстеження якості в реальному часі

Етичні та регуляторні аспекти

Ключові принципи:

- **Прозорість:** Лікарі повинні розуміти, як система приймає рішення
- **Підзвітність:** Чітка відповідальність за рішення системи
- **Справедливість:** Однакова якість для всіх демографічних груп
- **Конфіденційність:** Захист медичних даних пацієнтів
- **Безпека:** Мінімізація ризиків від помилкових діагнозів

Метрики якості для медичного застосування

$$\text{NPV (Negative Predictive Value)} = \frac{TN}{TN + FN} \tag{8}$$

$$\text{PPV (Positive Predictive Value)} = \frac{TP}{TP + FP} \tag{9}$$

$$\text{Diagnostic Odds Ratio} = \frac{TP \cdot TN}{FP \cdot FN} \tag{10}$$

$$\text{Number Needed to Diagnose} = \frac{1}{\text{Sensitivity} - (1 - \text{Specificity})} \tag{11}$$

Фінальна інтеграція системи

```
class ParkinsonDiagnosticSystem:
    def __init__(self):
        self.data_processor = SpiralDataProcessor()
        self.feature_extractor = FeatureExtractor()
        self.classifier = None  # Найкраща модель з ПР№3
        self.explainer = None
        self.validator = None

    def load_trained_models(self, model_path):
        """Завантаження навчених моделей"""
```

```
        # Завантаження збережених моделей
        pass

    def process_new_patient(self, spiral_data, patient_metadata):
        """Обробка нового пацієнта"""
        # 1. Обробка даних (ПРЖ1)
        processed_data = self.data_processor.preprocess(spiral_data)

        # 2. Видобування ознак (ПРЖ1-2)
        features = self.feature_extractor.extract_all_features(processed_data)

        # 3. Класифікація (ПРЖ3)
        prediction = self.classifier.predict_proba([features])[0]

        # 4. Генерація пояснень (ПРЖ4)
        report = self.explainer.generate_patient_report(
            features, patient_metadata['id']
        )

        return {
            'prediction': prediction,
            'report': report,
            'confidence': self.calculate_confidence(features, prediction),
            'recommendations': self.clinical_recommendations(prediction, report)
        }

    def clinical_recommendations(self, prediction, report):
        """Клінічні рекомендації на основі результатів"""
        risk_score = prediction[1]

        if risk_score > 0.8:
            return [
                "Рекомендується негайна консультація невролога",
                "Проведення додаткових неврологічних тестів",
                "Моніторинг моторних функцій"
            ]
        elif risk_score > 0.5:
            return [
                "Планова консультація невролога",
                "Повторне тестування через 6 місяців",
```

```
            "Спостереження за симптомами"
        ]
    else:
        return [
            "Результати в межах норми",
            "Рутинний огляд через рік",
            "При появі симптомів - додаткове обстеження"
        ]
```

Очікувані результати

1. Повністю валідована система діагностики хвороби Паркінсона
2. Комплексний аналіз продуктивності та обмежень системи
3. Інтерпретований ШІ з поясненнями для лікарів
4. Система готова до клінічних випробувань
5. Документація для регуляторного схвалення
6. Рекомендації для подальшого розвитку та покращення

Підсумковий звіт проєкту

Студенти повинні підготувати комплексний звіт, що включає:

- Опис повного циклу розробки системи
- Результати всіх етапів валідації
- Аналіз обмежень та можливостей покращення
- Етичні міркування та рекомендації
- План впровадження в клінічну практику

Контрольні питання

1. Які виклики виникають при валідації медичних систем ШІ?
2. Як забезпечити справедливість алгоритмів для різних демографічних груп?
3. Яка роль інтерпретованості в медичній діагностиці?
4. Які регуляторні вимоги застосовуються до медичних пристроїв на базі ШІ?
5. Як можна покращити довіру лікарів до систем штучного інтелекту?

Список джерел

[1] M. Kamble, P. Shrivastava та M. Jain, «Digitized spiral drawing classification for Parkinson's disease diagnosis,» *Measurement: Sensors*, т. 16, с. 100 047, 2021. DOI: 10.1016/j.measen.2021.100047 https://www.sciencedirect.com/science/article/pii/S266591742100009X.

[2] S. Aghanavesi, M. Memedi, M. Dougherty, D. Nyholm та J. Westin, «Verification of a Method for Measuring Parkinson's Disease Related Temporal Irregularity in Spiral Drawings,» *Sensors*, т. 17, № 10, с. 2341, 2017. DOI: 10.3390/s17102341 https://www.mdpi.com/1424-8220/17/10/2341.

[3] X. Chen, H. Zhang, R. Gao та L. Qiu, «Deep learning for biomedical signal processing: A review,» *IEEE Reviews in Biomedical Engineering*, т. 12, с. 127—141, 2019. DOI: 10.1109/RBME.2018.2874889

[4] P. Drotar, J. Mekyska, I. Rektorova, L. Masarova, Z. Smekal та M. Faundez-Zanuy, «Evaluation of handwriting kinematics and pressure for differential diagnosis of Parkinson's disease,» *Artificial intelligence in medicine*, т. 67, с. 39—46, 2016. DOI: 10.1016/j.artmed.2016.01.004

[5] J. P. Pinto, T. Pereira, A. Gonçalves, S. Lopes, M. Pinto, P. Rocha та ін., «Handwriting dynamics assessment through convolutional neural networks: An application to Parkinson's disease identification,» в *2019 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2019, с. 1—8. DOI: 10.1109/IJCNN.2019.8852226

[6] L. Breiman, «Random forests,» *Machine learning*, т. 45, № 1, с. 5—32, 2001. DOI: 10.1023/A:1010933404324

[7] D. S. Char, N. H. Shah та D. Magnus, «Implementing machine learning in health care—addressing ethical challenges,» *New England Journal of Medicine*, т. 378, № 11, с. 981—983, 2018. DOI: 10.1056/NEJMp1714229

[8] M. Isenkul, B. E. Sakar та O. Kursun, *Parkinson Disease Spiral Drawings Using Digitized Graphics Tablet*, UCI Machine Learning Repository, DOI: https://doi.org/10.24432/C5Q01S, 2013. https://archive.ics.uci.edu/dataset/395/parkinson+disease+spiral+drawings+using+digitized+graphics+tablet.

Додаток А. Приклад оформлення звіту практичної роботи

Титульний аркуш звіту

Звіт про виконання практичної роботи повинен містити титульний аркуш наступного формату:

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Тернопільський національний технічний університет
імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем
та програмної інженерії

Кафедра програмної інженерії

ЗВІТ

про виконання практичної роботи № __

з дисципліни

«Ідентифікація складних систем і об'єктів»

Тема: _____

Виконав(ла): _____
_____
____

Перевірив(ла): _____
____

Тернопіль — 2025

Рекомендована структура звіту

Звіт повинен містити такі основні розділи:

1. Вступ

- Мета виконання практичної роботи
- Актуальність проблеми, що розглядається
- Задачі роботи (перелік із нумерацією)

2. Теоретичні основи

- Короткий виклад теоретичних концепцій, необхідних для розв'язання задачі
- Основні визначення та формули
- Огляд використаних методів

3. Практична реалізація

- Описання алгоритму розв'язання
- Вихідні код (або посилання на Jupyter notebook в репозиторії)
- Технічні особливості реалізації
- Використані бібліотеки та інструменти

4. Результати та аналіз

- Результати експериментів
- Візуалізація (графіки, таблиці, діаграми)
- Інтерпретація результатів
- Порівняння з очікуваними результатами

5. Висновки

- Підсумування основних результатів
- Відповідь на поставлені в роботі питання
- Рекомендації для подальших досліджень
- Аналіз труднощів, з якими довелося мати справу

6. Список посилань

- Список всіх використаних джерел
- Рекомендовано використовувати стиль цитування IEEE
- Включити посилання на використані бібліотеки та датасети

Контрольний список для звіту

Перед здачею переконайтеся, що звіт містить:

- Титулка з усіма необхідними реквізитами
- Вступ з ясною постановкою задачі
- Теоретичні основи методів, що використовуються
- Опис алгоритму та реалізації
- Результати експериментів з візуалізацією
- Аналіз та інтерпретація результатів
- Висновки, що узагальнюють проведену роботу
- Список посилань на використані джерела
- Послідовність сторінок відповідно до змісту
- Правильне оформлення таблиць, рисунків та рівнянь
- Послідовність нумерації та посилань
- Відсутність граматичних та синтаксичних помилок
- Посилання на вихідний код в репозиторії