Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Фізико-технічний інститут

Захист програмного забезпечення

Розрахункова-графічна робота
«Біометрична автентифікація клавіатурного почерку»

Виконала:
Студентка групи ФБ-11
Данькова Єлізавета

Київ – 2024

Постановка задачі:

Розробити програму автентифікації користувача по клавіатурному почерку.

Вимоги до програми:

1. Програма повинна працювати в двох режимах:

- навчання (створення біометричного еталону)

- ідентифікації (порівняння з біометричним еталоном).

2. На етапі навчання необхідно визначати еталонні статистичні параметри клавіатурного почерку – оцінки математичного чекання і дисперсії тривалості утримання клавіш. Параметри повинні записуватися у файл. Вчення виробляти по багатократному набору фіксованій контрольній фразі, символи якої рівномірно розподілені по клавіатурі.

3. На етапі ідентифікації необхідно визначити параметри введеної контрольної фрази і перевірити гіпотезу про те, що отримані оцінки математичного очікування і дисперсії належать тому ж розподілу, що і параметри біометричного еталону. На цьому етапі необхідно відображувати отримані оцінки і еталонні параметри. Рівень значущості критерію задавати в діалоговому вікні.

4. На етапах навчання і ідентифікації передбачити можливість відбракування грубих помилок (окремих вимірів)

Опис математичного апарату та логіки програми:

Код реалізує систему автентифікації користувача за допомогою біометричних даних набору тексту на клавіатурі. Далі описано математичні методи та підходи, які використовуються в коді:

Математичний апарат:

1. **Статистичний аналіз:**
    - **Збір статистичних даних:** Коли користувач вводить фразу, зберігаються дані про швидкість набору, час утримання клавіш та динаміку натискання клавіш. Ці дані використовуються для обчислення середнього значення та стандартного відхилення.
    - **Оцінка інтервалів:** Для кожного з параметрів (швидкість набору, час утримання та динаміка натискання) обчислюються інтервали довіри на основі середнього значення та стандартного відхилення. Інтервали розраховуються як

      lower_bound = mean - 3 * std_deviation

      upper_bound = mean + 3 * std_deviation
    - **T-тест (Student's T-test)**

      Цей тест використовується для порівняння середнього значення нового набору даних зі значеннями, які вже є в системі:

      T-статистика (T-statistic) і p-значення (p-value) обчислюються для перевірки значимості відмінностей між новими і існуючими даними.

      $$t = \frac{\overline{X_1} - \overline{X_2}}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

      Де $\overline{X_1}$, $\overline{X_2}$ – середні значення двох груп даних, s_1, s_2 – стандартні відхилення груп, n_1, n_2 – розмір груп.
    - **F-тест (Fisher's F-test)**

      Цей тест використовується для порівняння дисперсій нового набору даних зі значеннями, які вже є в системі:

      F-статистика (F-statistic) і p-значення (p-value) обчислюються для перевірки значимості відмінностей між дисперсіями нових і існуючих даних.

$$F = \frac{s_1^2}{s_2^2}$$

2. **Машинне навчання:**
    - **Нормалізація даних:** Використовується StandardScaler зі sklearn для нормалізації даних перед навчанням моделі.
    - **Побудова моделі:** Використовується метод опорних векторів (Support Vector Regression, SVR) з лінійним ядром для побудови моделі. Модель навчається на зібраних біометричних даних.
    - **Крос-валідація:** Виконується крос-валідація з 5 фолдами для оцінки якості моделі. Крос-валідація дозволяє оцінити стабільність та узагальнюючу здатність моделі.
3. **Обробка динаміки натискання клавіш:**
    - **Вимірювання динаміки:** Вимірюються проміжки часу між натисканням клавіш під час введення фрази. Середній час між натисканнями клавіш використовується як показник динаміки.

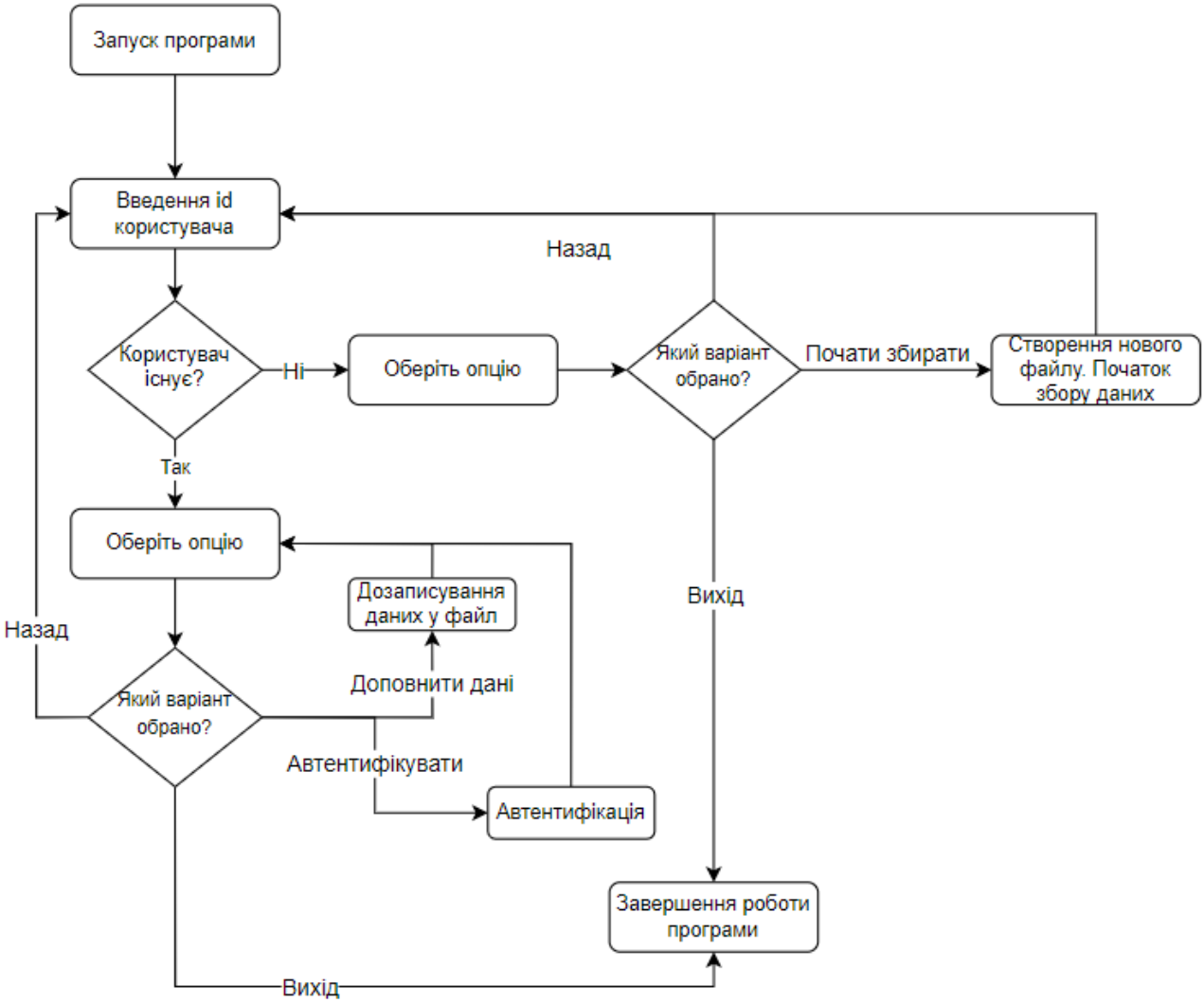
Процес:

1. **Збір даних:**
    - Користувач вводить задану кількість фраз.
    - Підраховується час утримання клавіш, швидкість набору та динаміка натискання клавіш.
    - Дані зберігаються у файл для подальшого використання.
2. **Побудова моделі:**
    - Дані нормалізуються.
    - Модель навчається на нормалізованих даних.
    - Виконується крос-валідація для оцінки моделі.
3. **Ідентифікація користувача:**
    - Користувач вводить фразу для ідентифікації.
    - Виміряні показники порівнюються з інтервалами довіри.
    - Виконується перевірка на відповідність даних користувача з раніше зібраними даними.
4. **Оновлення даних:**

- Існуючі дані можна доповнювати новими введеними фразами, що дозволяє постійно покращувати модель.

Код забезпечує можливість збирання даних, навчання моделі, виконання ідентифікації та доповнення існуючих даних, що дозволяє створювати та підтримувати систему автентифікації на основі біометричних характеристик набору тексту.

Блок схема роботи програми:



Робота програми:

```
Введіть ідентифікатор користувача: 1
Mean: 3.5577081022297707, Std Deviation: 0.8338366024729551
Lower Bound: -0.6114749101350045, Upper Bound: 7.726891114594546
Mean: 2.925566496167864, Std Deviation: 0.9518073129439268
Lower Bound: -1.8334700685517706, Upper Bound: 7.684603060887499
Mean: 0.10852284610456209, Std Deviation: 0.000822651897731614
Lower Bound: 0.10440958661590402, Upper Bound: 0.11263610559322015
Виберіть опцію:
1. Доповнити дані існуючого користувача
2. Автентифікувати існуючого
3. Назад
4. Вийти
```

```
Виберіть опцію:
1. Доповнити дані існуючого користувача
2. Автентифікувати існуючого
3. Назад
4. Вийти
1
Введіть фразу 36 для збору даних: ШІ ніколи не
Введіть фразу 37 для збору даних: замінить людей
Введіть фразу 38 для збору даних: повністю
Введіть фразу 39 для збору даних: завжди залишуться
Введіть фразу 40 для збору даних: ті хто нам потрібен
Mean: 3.5227156072994674, Std Deviation: 0.9302841939554759
Lower Bound: -1.1287053624779118, Upper Bound: 8.174136577076847
Mean: 3.387226539850235, Std Deviation: 2.6237378575917965
Lower Bound: -9.731462748108747, Upper Bound: 16.505915827809 22
Mean: 0.10852964018257216, Std Deviation: 0.0007805863106421177
Lower Bound: 0.10462670862936158, Upper Bound: 0.11243257173578275
```

```
Виберіть опцію:
1. Доповнити дані існуючого користувача
2. Автентифікувати існуючого
3. Назад
4. Вийти
2
Введіть фразу для ідентифікації: Я ніколи не полишу
Typing Speed - Lower Bound: -1.1287053624779118, Upper Bound Speed: 8.174136577076847
Hold Time - Lower Bound: -9.731462748108747, Upper Bound Hold: 16.50591582780922
Key Press Dynamics - Lower Bound: 0.10462670862936158, Upper Bound Dynamics: 0.11243257173578275
Hold Time: 5.5123937129974365, Typing Speed: 3.2653690823205483, Key Press Dynamics: 0.10858681622673483
Час утримання не є аномальним.
Швидкість набору не є аномальною.
Динаміка натискання клавіш не є аномальною.
t-statistic (Speed): 0.2698006808334052, p-value (Speed): 0.7887361002455764
t-statistic (Hold Time): -0.7899744265127805, p-value (Hold Time): 0.43431985673452156
t-statistic (Dynamics): -0.07143870100533839, p-value (Dynamics): 0.9434137063811474
Швидкість набору не суттєво відрізняється від наявних даних.
Час утримання не суттєво відрізняється від наявних даних.
Динаміка натискання клавіш не суттєво відрізняється від наявних даних.
F-statistic (Speed): 0.07279240737816904, p-value (Speed): 0.7887361002455784
F-statistic (Hold Time): 0.6240595945441963, p-value (Hold Time): 0.43431985673452
F-statistic (Dynamics): 0.005103488001328711, p-value (Dynamics): 0.9434137063811395
Дисперсія швидкості набору не суттєво відрізняється від наявних даних.
Дисперсія часу утримання не суттєво відрізняється від наявних даних.
Дисперсія динаміки натискання клавіш не суттєво відрізняється від наявних даних.
Користувача 1 автентифіковано
Швидкість набору: 3.2653690823205483
Speed difference 0.3091586911528914
Час утримання: 5.5123937129974365
Hold time difference: 0.7565207481384277
Динаміка натискання клавіш: 0.10858681622673483
```

```
Dynamics difference: 0.00031180504490346816
Виберіть опцію:
1. Доповнити дані існуючого користувача
2. Автентифікувати існуючого
3. Назад
4. Вийти
```

```
Виберіть опцію:
1. Доповнити дані існуючого користувача
2. Автентифікувати існуючого
3. Назад
4. Вийти
3
Введіть ідентифікатор користувача: 8
Такого користувача не існує. Бажаєте почати збирати дані?
1. Почати збирати
2. Назад до введення ідентифікатора
3. Вихід
```

```
Виберіть опцію:
1. Доповнити дані існуючого користувача
2. Автентифікувати існуючого
3. Назад
4. Вийти
3
Введіть ідентифікатор користувача: 8
Такого користувача не існує. Бажаєте почати збирати дані?
1. Почати збирати
2. Назад до введення ідентифікатора
3. Вихід
1
Введіть фразу 1 для збору даних: А сонце світить
Введіть фразу 2 для збору даних: Сонце світить всім
Введіть фразу 3 для збору даних: Однаково світить сонце
Введіть фразу 4 для збору даних: Завжди для нас
Введіть фразу 5 для збору даних: Вонон є
```

```
Mean: 3.775588402396906, Std Deviation: 0.9784400817727156
Lower Bound: -1.1166120064666716, Upper Bound: 8.667788811260483
Mean: 4.021481227874756, Std Deviation: 0.9314669396712485
Lower Bound: -0.6358534704814867, Upper Bound: 8.678815926231
Mean: 0.10895853294840252, Std Deviation: 0.0003861431964879983
Lower Bound: 0.10702781696596253, Upper Bound: 0.11088924893084251
Введіть ідентифікатор користувача:
```

```
Введіть ідентифікатор користувача: 10
Такого користувача не існує. Бажаєте почати збирати дані?
1. Почати збирати
2. Назад до введення ідентифікатора
3. Вихід
2
Введіть ідентифікатор користувача: 11
Такого користувача не існує. Бажаєте почати збирати дані?
1. Почати збирати
2. Назад до введення ідентифікатора
3. Вихід
3

Process finished with exit code 0
```

Додаток (повний код):

```python
import os
import time
import json
import numpy as np
from scipy import stats
from scipy.stats import ttest_ind, f_oneway
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVR
from sklearn.model_selection import cross_val_score

class KeyboardBiometrics:
    def __init__(self, training_iterations, user_id):
        self.training_iterations = training_iterations
        self.user_id = user_id
        self.biometric_data = []
        self.model = self.build_model()
        self.scaler = StandardScaler()
        self.typing_speed_stats = None
        self.hold_time_stats = None
        self.key_press_dynamics_stats = None

    def build_model(self):
        return SVR(kernel='linear')
```

```
    def collect_data(self):
        all_data = []
        for i in range(1, self.training_iterations + 1):
            start_time = time.time()
            phrase = input(f"Введіть фразу {i} для збору даних: ")
            end_time = time.time()
            hold_time = end_time - start_time
            typing_speed = len(phrase) / hold_time

            key_press_dynamics = self.measure_key_press_dynamics(phrase)

            data = {'user_id': self.user_id, 'phrase': phrase,
'phrase_number': i, 'typing_speed': typing_speed,
                    'hold_time': hold_time, 'key_press_dynamics':
key_press_dynamics}
            all_data.append(data)
            self.biometric_data.append([self.user_id, i, typing_speed,
hold_time, key_press_dynamics])

        filename = f'biometric_data_user_{self.user_id}.json'
        if not os.path.exists(filename):
            with open(filename, 'w') as file:
                json.dump(all_data, file, indent=4)
        else:
            with open(filename, 'r') as file:
                existing_data = json.load(file)
                existing_data.extend(all_data)
            with open(filename, 'w') as file:
                json.dump(existing_data, file, indent=4)

        biometric_data_array = np.array(self.biometric_data)
        self.scaler.fit(biometric_data_array[:, 2:])
        biometric_data_scaled = self.scaler.transform(biometric_data_array[:,
2:])

        X = biometric_data_scaled
        y = biometric_data_array[:, 3]
        scores = cross_val_score(self.model, X, y, cv=5)
        print("Cross-Validation Scores:", scores)
        print("Mean CV Score:", np.mean(scores))

        self.model.fit(X, y)
        self.calculate_intervals()

    def measure_key_press_dynamics(self, phrase):
        key_press_times = []
        for _ in phrase:
            key_press_times.append(time.time())
            time.sleep(0.1)

        avg_key_press_time = np.mean(np.diff(key_press_times)) if
len(key_press_times) > 1 else 0
        return avg_key_press_time

    def identify(self):
        start_time = time.time()
        phrase = input("Введіть фразу для ідентифікації: ")
        end_time = time.time()
        hold_time = end_time - start_time
        typing_speed = len(phrase) / hold_time
        key_press_dynamics = self.measure_key_press_dynamics(phrase)

        lower_bound_speed, upper_bound_speed = self.typing_speed_stats
```

```
        lower_bound_hold, upper_bound_hold = self.hold_time_stats
        lower_bound_dynamics, upper_bound_dynamics =
self.key_press_dynamics_stats

        print(f"Typing Speed - Lower Bound: {lower_bound_speed}, Upper Bound
Speed: {upper_bound_speed}")
        print(f"Hold Time - Lower Bound: {lower_bound_hold}, Upper Bound
Hold: {upper_bound_hold}")
        print(f"Key Press Dynamics - Lower Bound: {lower_bound_dynamics},
Upper Bound Dynamics: {upper_bound_dynamics}")
        print(f"Hold Time: {hold_time}, Typing Speed: {typing_speed}, Key
Press Dynamics: {key_press_dynamics}")

        if lower_bound_hold <= hold_time <= upper_bound_hold:
            print("Час утримання не є аномальним.")
        else:
            print("Час утримання є аномальним.")

        if lower_bound_speed <= typing_speed <= upper_bound_speed:
            print("Швидкість набору не є аномальною.")
        else:
            print("Швидкість набору є аномальною.")

        if lower_bound_dynamics <= key_press_dynamics <=
upper_bound_dynamics:
            print("Динаміка натискання клавіш не є аномальною.")
        else:
            print("Динаміка натискання клавіш є аномальною.")

        new_data = {
            'typing_speed': typing_speed,
            'hold_time': hold_time,
            'key_press_dynamics': key_press_dynamics
        }

        self.t_test_new_data(new_data)
        self.f_test_new_data(new_data)

        filename = f'biometric_data_user_{self.user_id}.json'
        if os.path.exists(filename):
            with open(filename, 'r') as file:
                all_data = json.load(file)

                for data in all_data:
                    saved_typing_speed = data['typing_speed']
                    saved_hold_time = data['hold_time']
                    saved_key_press_dynamics = data['key_press_dynamics']

                    threshold_hold_time = 1
                    threshold_typing_speed = 1
                    threshold_key_press_dynamics = 0.1

                    hold_time_difference = abs(hold_time - saved_hold_time)
                    speed_difference = abs(typing_speed - saved_typing_speed)
                    dynamics_difference = abs(key_press_dynamics -
saved_key_press_dynamics)

                    if (hold_time_difference < threshold_hold_time and
                            speed_difference < threshold_typing_speed and
                            dynamics_difference <
threshold_key_press_dynamics):
                        print(f"Користувача {self.user_id} автентифіковано")
                        print("Швидкість набору:", typing_speed)
                        print("Speed difference", speed_difference)
```

```
                        print("Час утримання:", hold_time)
                        print("Hold time difference:", hold_time_difference)
                        print("Динаміка натискання клавіш:",
key_press_dynamics)
                        print("Dynamics difference:", dynamics_difference)
                        return

                print(f"Користувач {self.user_id} - не користувач.")
        else:
            print(f"Файл з даними для користувача {self.user_id} не існує.")

    def calculate_intervals(self):
        filename = f'biometric_data_user_{self.user_id}.json'
        intervals_speed, intervals_hold, intervals_dynamics = [], [], []

        if os.path.exists(filename):
            with open(filename, 'r') as file:
                all_data = json.load(file)
                for data in all_data:
                    intervals_speed.append(data['typing_speed'])
                    intervals_hold.append(data['hold_time'])
                    intervals_dynamics.append(data['key_press_dynamics'])

            self.typing_speed_stats = self.calculate_bounds(intervals_speed)
            self.hold_time_stats = self.calculate_bounds(intervals_hold)
            self.key_press_dynamics_stats =
self.calculate_bounds(intervals_dynamics)
        else:
            print(f"Файл з даними для користувача {self.user_id} не існує.")

    def calculate_bounds(self, intervals):
        mean = np.mean(intervals)
        std_deviation = np.std(intervals)
        lower_bound = mean - 5 * std_deviation
        upper_bound = mean + 5 * std_deviation

        print(f"Mean: {mean}, Std Deviation: {std_deviation}")
        print(f"Lower Bound: {lower_bound}, Upper Bound: {upper_bound}")

        return lower_bound, upper_bound

    def t_test_new_data(self, new_data):
        filename = f'biometric_data_user_{self.user_id}.json'
        if os.path.exists(filename):
            with open(filename, 'r') as file:
                all_data = json.load(file)
                typing_speeds = [data['typing_speed'] for data in all_data]
                hold_times = [data['hold_time'] for data in all_data]
                key_press_dynamics = [data['key_press_dynamics'] for data in
all_data]

            t_stat_speed, p_val_speed = ttest_ind(typing_speeds,
[new_data['typing_speed']])
            t_stat_hold, p_val_hold = ttest_ind(hold_times,
[new_data['hold_time']])
            t_stat_dynamics, p_val_dynamics = ttest_ind(key_press_dynamics,
[new_data['key_press_dynamics']])

            print(f"t-statistic (Speed): {t_stat_speed}, p-value (Speed):
{p_val_speed}")
            print(f"t-statistic (Hold Time): {t_stat_hold}, p-value (Hold
Time): {p_val_hold}")
            print(f"t-statistic (Dynamics): {t_stat_dynamics}, p-value
(Dynamics): {p_val_dynamics}")
```

```
            if p_val_speed < 0.05:
                print("Швидкість набору суттєво відрізняється від наявних
даних.")
            else:
                print("Швидкість набору не суттєво відрізняється від наявних
даних.")

            if p_val_hold < 0.05:
                print("Час утримання суттєво відрізняється від наявних
даних.")
            else:
                print("Час утримання не суттєво відрізняється від наявних
даних.")

            if p_val_dynamics < 0.05:
                print("Динаміка натискання клавіш суттєво відрізняється від
наявних даних.")
            else:
                print("Динаміка натискання клавіш не суттєво відрізняється
від наявних даних.")

    def f_test_new_data(self, new_data):
        filename = f'biometric_data_user_{self.user_id}.json'
        if os.path.exists(filename):
            with open(filename, 'r') as file:
                all_data = json.load(file)
                typing_speeds = [data['typing_speed'] for data in all_data]
                hold_times = [data['hold_time'] for data in all_data]
                key_press_dynamics = [data['key_press_dynamics'] for data in
all_data]

            f_stat_speed, p_val_speed = f_oneway(typing_speeds,
[new_data['typing_speed']])
            f_stat_hold, p_val_hold = f_oneway(hold_times,
[new_data['hold_time']])
            f_stat_dynamics, p_val_dynamics = f_oneway(key_press_dynamics,
[new_data['key_press_dynamics']])

            print(f"F-statistic (Speed): {f_stat_speed}, p-value (Speed):
{p_val_speed}")
            print(f"F-statistic (Hold Time): {f_stat_hold}, p-value (Hold
Time): {p_val_hold}")
            print(f"F-statistic (Dynamics): {f_stat_dynamics}, p-value
(Dynamics): {p_val_dynamics}")

            if p_val_speed < 0.05:
                print("Дисперсія швидкості набору суттєво відрізняється від
наявних даних.")
            else:
                print("Дисперсія швидкості набору не суттєво відрізняється
від наявних даних.")

            if p_val_hold < 0.05:
                print("Дисперсія часу утримання суттєво відрізняється від
наявних даних.")
            else:
                print("Дисперсія часу утримання не суттєво відрізняється від
наявних даних.")

            if p_val_dynamics < 0.05:
                print("Дисперсія динаміки натискання клавіш суттєво
відрізняється від наявних даних.")
            else:
```

```
                print("Дисперсія динаміки натискання клавіш не суттєво
відрізняється від наявних даних.")

    def save_additional_data(self):
        filename = f'biometric_data_user_{self.user_id}.json'
        all_data = []

        if os.path.exists(filename):
            with open(filename, 'r') as file:
                all_data = json.load(file)

            last_data_entry = all_data[-1] if all_data else None

            if last_data_entry:
                start_phrase_number = last_data_entry.get('phrase_number', 0)
+ 1
            else:
                start_phrase_number = 1
        else:
            start_phrase_number = 1

        new_data = []
        for i in range(start_phrase_number, start_phrase_number +
self.training_iterations):
            start_time = time.time()
            phrase = input(f"Введіть фразу {i} для збору даних: ")
            end_time = time.time()
            hold_time = end_time - start_time
            typing_speed = len(phrase) / hold_time

            key_press_dynamics = self.measure_key_press_dynamics(phrase)

            data = {'user_id': self.user_id, 'phrase': phrase,
'phrase_number': i, 'typing_speed': typing_speed,
                    'hold_time': hold_time, 'key_press_dynamics':
key_press_dynamics}
            new_data.append(data)

        all_data.extend(new_data)

        with open(filename, 'w') as file:
            json.dump(all_data, file, indent=4)

        self.calculate_intervals()

def main():
    training_iterations = 5

    while True:
        user_id = input("Введіть ідентифікатор користувача: ")
        filename = f'biometric_data_user_{user_id}.json'

        if not os.path.exists(filename):
            option = input("Такого користувача не існує. Бажаєте почати
збирати дані?\n1. Почати збирати\n2. Назад до введення ідентифікатора\n3.
Вихід\n")
            if option == "1":
                biometrics = KeyboardBiometrics(training_iterations, user_id)
                biometrics.collect_data()
            elif option == "2":
                continue
            elif option == "3":
                break
            else:
```

```
                print("Неправильний вибір опції.")
        else:
            biometrics = KeyboardBiometrics(training_iterations, user_id)
            biometrics.calculate_intervals()
            while True:
                option = input("Виберіть опцію:\n1. Доповнити дані існуючого
користувача\n2. Автентифікувати існуючого\n3. Назад\n4. Вийти\n")
                if option == "1":
                    biometrics.save_additional_data()
                elif option == "2":
                    biometrics.identify()
                elif option == "3":
                    break
                elif option == "4":
                    return
                else:
                    print("Неправильний вибір опції.")

if __name__ == "__main__":
    main()
```