รายงานวิชา 261456

เรื่อง

Multilayer Perceptron (MLP) using Particle Swarm optimization (PSO)

จัดทำโดย

นายธนัช ธนัญชัย 650610838

เสนอ

รศ.ดร.ศันสนีย์ เอื้อพันธ์วิริยะกุล

รายงานนนี้เป็นส่วนหนึ่งของวิชา 261456

ภาคการศึกษาที่ 1 ปีการศึกษา 2567

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเชียงใหม่

คำนำ

รายงานฉบับนี้เป็นส่วนหนึ่งของวิชา 261456 โดยมีวัตถุประสงค์เพื่อพัฒนาและฝึกสอนโครงข่ายประสาทเทียมแบบหลายชั้น (MLP) สำหรับการจำแนกประเภทข้อมูลโดยใช้การทำ Particle Swarm Optimization (PSO) คือการหาคำตอบที่เหมาะสมที่สุดของปัญหาการหาค่าเหมาะสม (Optimization Problem) โดยใช้แนวคิดของการเคลื่อนที่ของกลุ่มอนุภาค (Particles) ที่เลียนแบบพฤติกรรมการหาทิศทางของฝูงนกหรือปลาผ่านการเคลื่อนที่ไปยังตำแหน่งที่ดีกว่าเรื่อย ๆ ในแต่ละรอบของการคำนวณ โดย PSO มุ่งเน้นในการค้นหาค่าที่เหมาะสมที่สุดของฟังก์ชันที่กำหนด โดยให้อนุภาคแต่ละตัวมีการเรียนรู้จากตำแหน่งที่ดีของตัวเองและตำแหน่งที่ดีของเพื่อนร่วมฝูง

ผู้จัดทำ

27/10/2567

ผลการทดลอง

จากการทดลองในการเขียนโปรแกรมสำหรับการ Train Multilayer Perceptron โดยใช้ Particle Swarm Optimization (PSO) สำหรับการทำ prediction Benzene concentration โดยเป็นการ predict 5 วันล่วงหน้า และ 10 วันล่วงหน้า โดยให้ใช้ attribute เบอร์ 3,6,8,10,11,12,13 และ 14 เป็น input ส่วน desire output เป็น attribute เบอร์ 5 ให้ทำการทดลองกับ AirQualityUCI (Air Quality Data Set จาก UCI Machine learning Repository) โดยที่ data set นี้มีทั้งหมด 9358 sample และมี 14 attribute ดังนี้

0 Date (DD/MM/YYYY)

1 Time (HH.MM.SS)

2 True hourly averaged concentration CO in mg/m^3 (reference analyzer)

3 PT08.S1 (tin oxide) hourly averaged sensor response (nominally CO targeted)

4 True hourly averaged overall Non Metanic HydroCarbons concentration in microg/m^3 (reference analyzer)

5 True hourly averaged Benzene concentration in microg/m^3 (reference analyzer)

6 PT08.S2 (titania) hourly averaged sensor response (nominally NMHC targeted)

7 True hourly averaged NOx concentration in ppb (reference analyzer)

8 PT08.S3 (tungsten oxide) hourly averaged sensor response (nominally NOx targeted)

9 True hourly averaged NO2 concentration in microg/m^3 (reference analyzer)

10 PT08.S4 (tungsten oxide) hourly averaged sensor response (nominally NO2 targeted)

11 PT08.S5 (indium oxide) hourly averaged sensor response (nominally O3 targeted)

12 Temperature in Â°C

13 Relative Humidity (%)

14 AH Absolute Humidity

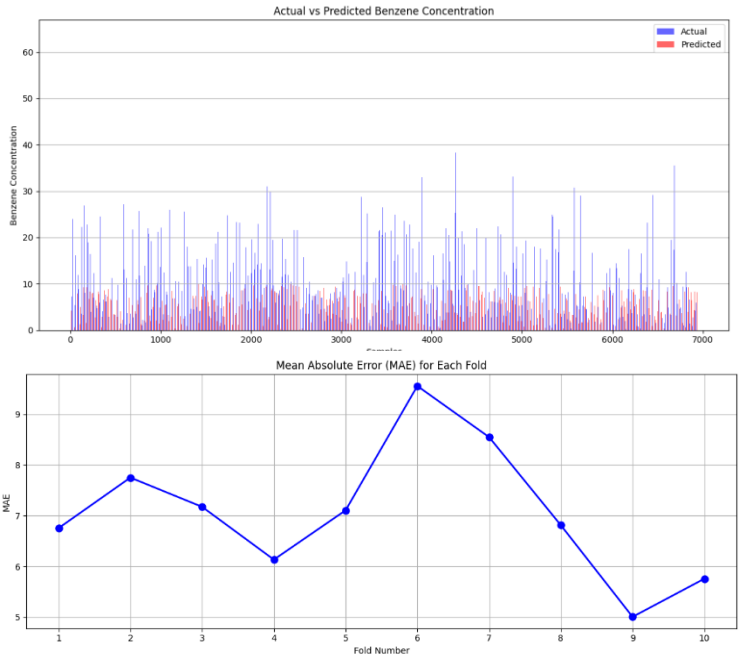
ให้ทำการทดลองโดยใช้ 10% cross validation เพื่อทดสอบ validity ของ network ที่ได้ และให้ทำการเปลี่ยนแปลงจำนวน hidden layer และ nodes

ในการวัด Error ให้ใช้ **Mean Absolute Error (MAE)** ได้ข้อสรุปดังนี้

ทดลองโดยมีการจัดค่าพารามิเตอร์ทั้งหมดดังนี้

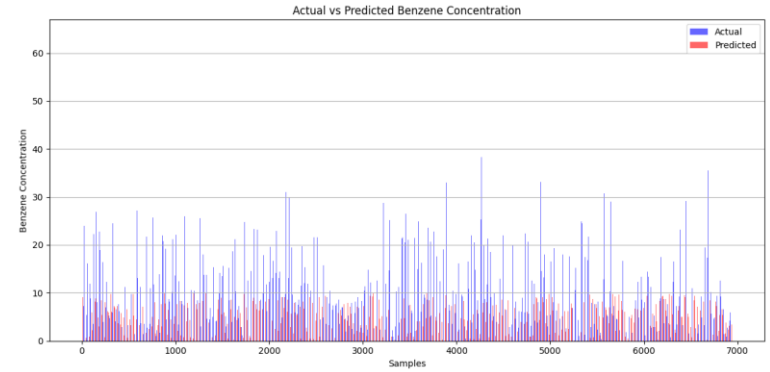
กรณีเปลี่ยน particles

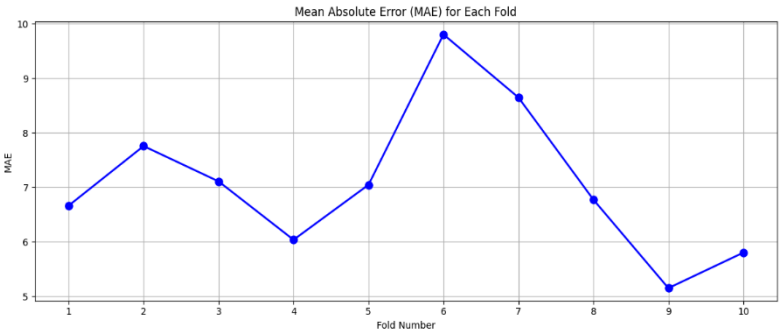
กำหนดให้ particles = 50 , hidden layers = 15 , hidden nodes = 10 , iterations = 100 ได้ผลลัพธ์ออกมาดังรูป




```
MAE for Fold 1: 6.756153054380759
MAE for Fold 2: 7.751147306692675
MAE for Fold 3: 7.174520013367939
MAE for Fold 4: 6.134676555874204
MAE for Fold 5: 7.106079535150188
MAE for Fold 6: 9.55574561449324
MAE for Fold 7: 8.548481372691654
MAE for Fold 8: 6.813047581363406
MAE for Fold 9: 5.006966615101494
MAE for Fold 10: 5.757711212356892
Overall MAE across folds: 7.060452886147244
```

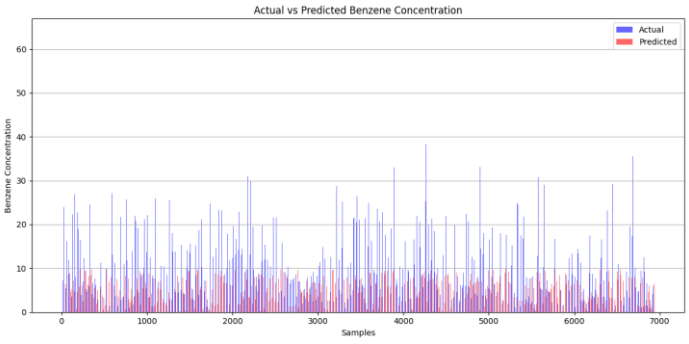
กำหนดให้ particles = 100 , hidden layers = 15 , hidden nodes = 10 , iterations = 100 ได้ผลลัพธ์ออกมาดังรูป

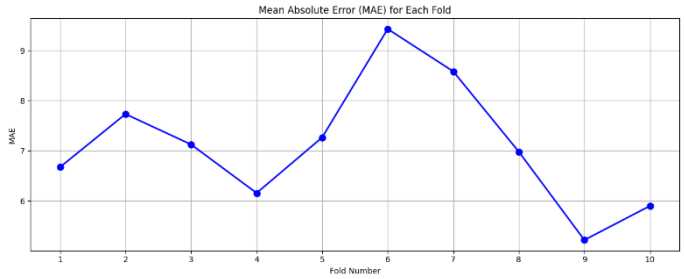





```
MAE for Fold 1: 6.661404279688423
MAE for Fold 2: 7.754510794004852
MAE for Fold 3: 7.106441338439125
MAE for Fold 4: 6.034912817315559
MAE for Fold 5: 7.043809383106923
MAE for Fold 6: 9.802310174553728
MAE for Fold 7: 8.645448670215023
MAE for Fold 8: 6.771183814807214
MAE for Fold 9: 5.148400716497259
MAE for Fold 10: 5.7998200087990295
Overall MAE across folds: 7.076824199742714
```

กำหนดให้ particles = 200 , hidden layers = 15 , hidden nodes = 10 , iterations = 100 ได้ผลลัพธ์ออกมาดังรูป

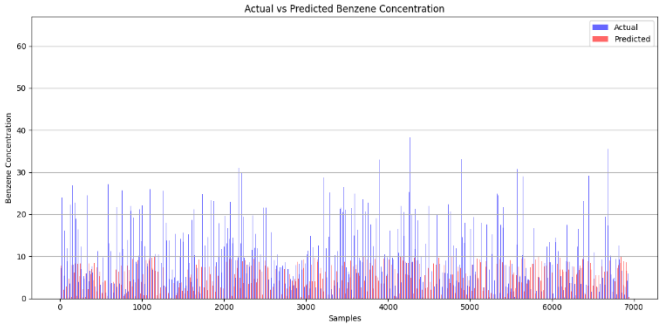


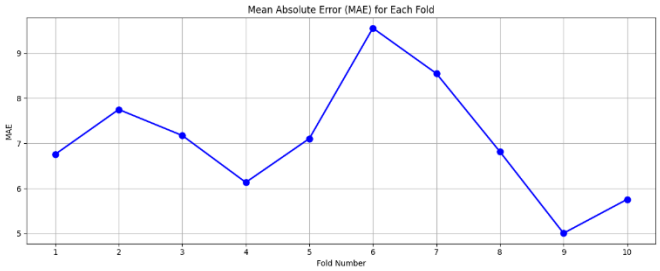



```
MAE for Fold 1: 6.673321258804316
MAE for Fold 2: 7.732007899501916
MAE for Fold 3: 7.126089355065487
MAE for Fold 4: 6.1582335097396665
MAE for Fold 5: 7.266646959459843
MAE for Fold 6: 9.426948274020376
MAE for Fold 7: 8.581688182182805
MAE for Fold 8: 6.980818735028824
MAE for Fold 9: 5.222496772144304
MAE for Fold 10: 5.9011640494607125
Overall MAE across folds: 7.1069414995408255
```

กรณีเปลี่ยน hidden layers

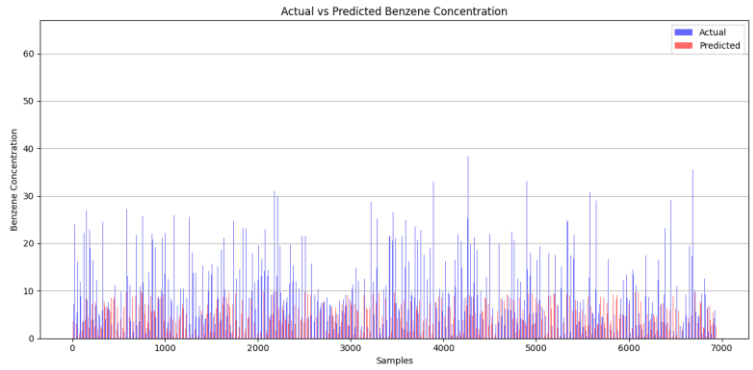
กำหนดให้ particles = 200 , hidden layers = 10 , hidden nodes = 10 , iterations = 100 ได้ผลลัพธ์ออกมาดังรูป

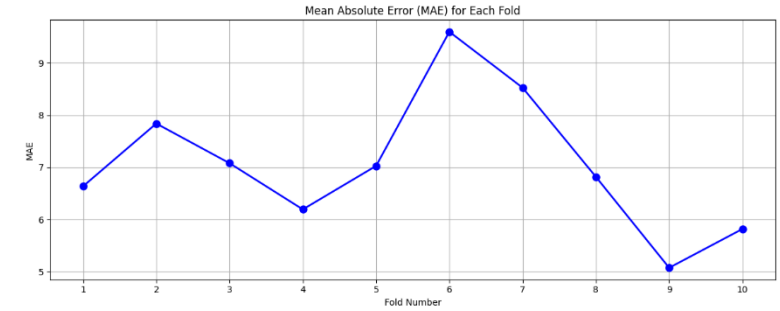





```
MAE for Fold 1: 6.7160110027797
MAE for Fold 2: 7.8169830577957296
MAE for Fold 3: 7.264007792032943
MAE for Fold 4: 6.116690879896305
MAE for Fold 5: 7.192604295570459
MAE for Fold 6: 9.56619759049399
MAE for Fold 7: 8.436452526524976
MAE for Fold 8: 6.932435282304277
MAE for Fold 9: 5.186025997747716
MAE for Fold 10: 5.830605275935725
Overall MAE across folds: 7.105801370108182
```

กำหนดให้ particles = 200 , hidden layers = 30 , hidden nodes = 10 , iterations = 100 ได้ผลลัพธ์ออกมาดังรูป

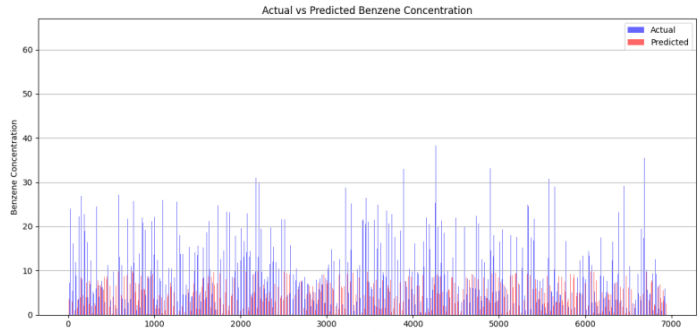


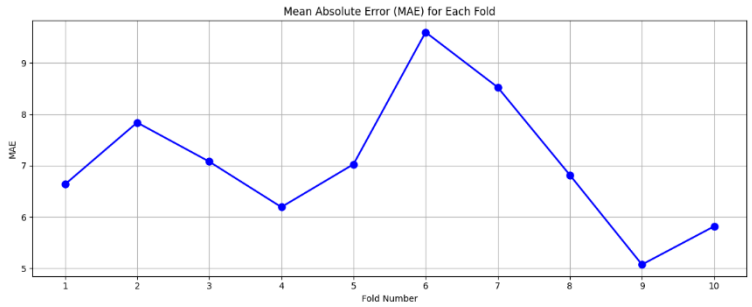



```
MAE for Fold 1: 6.6354868163640175
MAE for Fold 2: 7.839331872213663
MAE for Fold 3: 7.078659561987382
MAE for Fold 4: 6.193060689629179
MAE for Fold 5: 7.02685985477284
MAE for Fold 6: 9.598538337340113
MAE for Fold 7: 8.525901061667922
MAE for Fold 8: 6.817560865152304
MAE for Fold 9: 5.07365859658263
MAE for Fold 10: 5.817569016008254
Overall MAE across folds: 7.06066266717183
```

กรณีเปลี่ยน iterations

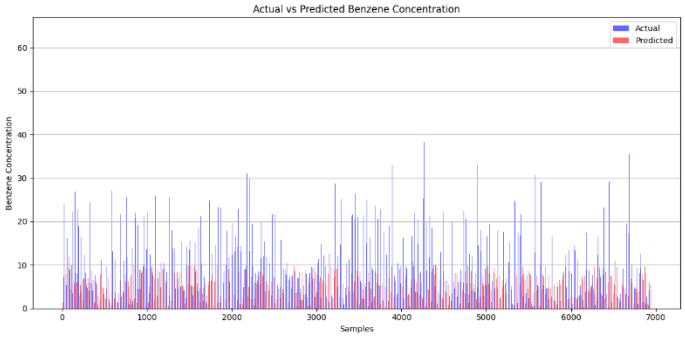
กำหนดให้ particles = 200 , hidden layers = 30 , hidden nodes = 10 , iterations = 100 ได้ผลลัพธ์ออกมาดังรูป

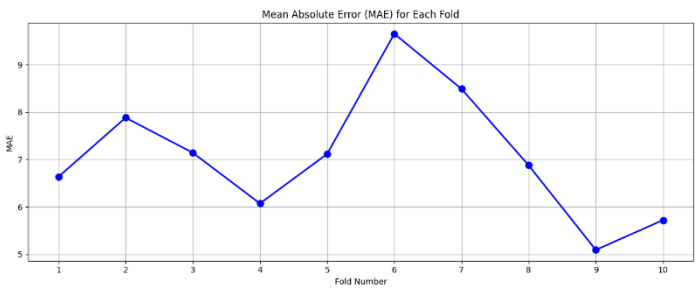





```
MAE for Fold 1: 6.6354868163640175
MAE for Fold 2: 7.839331872213663
MAE for Fold 3: 7.078659561987382
MAE for Fold 4: 6.193060689629179
MAE for Fold 5: 7.02685985477284
MAE for Fold 6: 9.598538337340113
MAE for Fold 7: 8.525901061667922
MAE for Fold 8: 6.817560865152304
MAE for Fold 9: 5.07365859658263
MAE for Fold 10: 5.817569016008254
Overall MAE across folds: 7.06066266717183
```

กำหนดให้ particles = 200 , hidden layers = 30 , hidden nodes = 10 , iterations = 500 ได้ผลลัพธ์ออกมาดังรูป






```
MAE for Fold 1: 6.634148736583662
MAE for Fold 2: 7.881108702774889
MAE for Fold 3: 7.141414553584215
MAE for Fold 4: 6.0716496834289755
MAE for Fold 5: 7.115287692356711
MAE for Fold 6: 9.650879964211478
MAE for Fold 7: 8.48887676368859
MAE for Fold 8: 6.876747764092427
MAE for Fold 9: 5.091900082275485
MAE for Fold 10: 5.721238454601785
Overall MAE across folds: 7.067325239759822
```

วิเคราะห์ผลการทดลอง

จากการทดลองข้างต้นแบ่งเป็นประเด็นสำคัญหลักๆได้ดังนี้

1. MAE แต่ละโฟลด์

MAE (Mean Absolute Error) คือการวัดความแตกต่างระหว่างค่าที่คาดการณ์และค่าจริงในชุดข้อมูล โดยค่าที่ต่ำกว่าจะบ่งบอกว่าโมเดลคาดการณ์ได้แม่นยำ ในขณะที่ค่าที่สูงอาจบ่งชี้ถึงความไม่แม่นยำของโมเดล ซึ่งอาจเกิดจากหลายปัจจัย เช่น การกระจายตัวของข้อมูลที่สูง, ความไม่สามารถของโมเดลในการจับข้อมูลที่ซับซ้อน, หรือปัญหาจากคุณภาพข้อมูล เช่น ค่าผิดปกติหรือข้อมูลที่ขาดหายไป

2. MAE รวม

MAE รวม คือค่าเฉลี่ยของ MAE จากทุกโฟลด์ ซึ่งเป็นตัวแทนประสิทธิภาพของโมเดลในการคาดการณ์โดยรวม หากค่า MAE รวมสูง จะบ่งชี้ว่าโมเดลต้องการการปรับปรุง เช่น การปรับค่าพารามิเตอร์หรือการเปลี่ยนแปลงโครงสร้างโมเดลเพื่อเพิ่มความแม่นยำในการคาดการณ์.

3. การสร้างภาพ

กราฟ MAE แสดงค่า MAE ของแต่ละโฟลด์ ซึ่งช่วยให้เห็นว่ามีโฟลด์ใดที่โมเดลทำได้ดีหรือไม่ดี ในขณะที่กราฟ Actual vs. Predicted แสดงค่าจริง (Actual) และค่าที่คาดการณ์ (Predicted) เพื่อช่วยในการเปรียบเทียบความแม่นยำของโมเดล โดยค่าที่คาดการณ์ใกล้เคียงค่าจริงจะแสดงว่าโมเดลทำงานได้ดี และยังสามารถใช้เพื่อตรวจสอบค่าผิดปกติที่อาจชี้ให้เห็นถึงปัญหาในข้อมูลหรือโมเดลได้อีกด้วย

สรุปการใช้ MAE เป็นเครื่องมือวัดความแม่นยำของโมเดลทั้งในแต่ละโฟลด์และในระดับรวม ช่วยให้เห็นภาพรวมของประสิทธิภาพได้ชัดเจนขึ้น โดยกราฟที่แสดงผลการทำงานช่วยให้เข้าใจข้อมูลและการคาดการณ์ได้ดีขึ้น ทำให้สามารถตัดสินใจในการปรับปรุงโมเดลในอนาคตได้อย่างมีประสิทธิภาพ การวิเคราะห์ผลนี้ช่วยให้สามารถปรับปรุงโมเดลได้อย่างเหมาะสมตามข้อบกพร่องที่พบในแต่ละโฟลด์และปรับแต่งการทำงานของโมเดลให้มีประสิทธิภาพมากขึ้นในการคาดการณ์ความเข้มข้นของเบนซินในชุดข้อมูลนี้

ลิงก์โปรแกรม PONDZA566/CI_4-Particle-Swarm-Optimization

ตัวอย่างโค้ด

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Set a fixed random seed
np.random.seed(42)

# Load dataset
data = pd.read_excel('AirQualityUCI.xlsx', decimal=',', na_values=-200)

# Drop rows with NaN values
data = data.dropna()

# Select attributes
X = data.iloc[:, [3, 6, 8, 10, 11, 12, 13]].values  # input attributes
y = data.iloc[:, 5].values  # output attribute (Benzene concentration)

# Normalize data
X = (X - np.mean(X, axis=0)) / np.std(X, axis=0)

# Function to predict benzene concentration (using a mock prediction for now)
def predict_benzene(X, best_hyperparams):
    return np.random.rand(len(X)) * 10  # Simulating predictions

# Function to calculate MAE
def mean_absolute_error(y_true, y_pred):
    return np.mean(np.abs(y_true - y_pred))
```

```python
# Define a function for cross-validation
def cross_validate(X, y, n_folds=10):
    fold_size = len(X) // n_folds
    mae_per_fold = []
    all_predictions = np.zeros_like(y)

    for fold in range(n_folds):
        test_indices = range(fold * fold_size, (fold + 1) * fold_size)
        X_train = np.delete(X, test_indices, axis=0)
        y_train = np.delete(y, test_indices)
        X_test = X[test_indices]
        y_test = y[test_indices]

        # Train the MLP and get predictions
        hyperparams = best_hyperparams  # Assuming hyperparams are already optimized
        predictions = predict_benzene(X_test, hyperparams)

        all_predictions[test_indices] = predictions

        # Calculate MAE and store it
        mae = mean_absolute_error(y_test, predictions)
        mae_per_fold.append(mae)

    return mae_per_fold, all_predictions

# Mock function to simulate training process
def train_mlp(hyperparams, X_train, y_train):
    return 1.5  # Mock MAE value
```

```python
# Define PSO for optimizing MLP
class Particle:
    def __init__(self, n_hidden_layers, n_nodes):
        self.position = np.random.rand(n_hidden_layers, n_nodes)
        self.velocity = np.random.rand(n_hidden_layers, n_nodes) * 0.1
        self.best_position = self.position.copy()
        self.best_error = float('inf')

def pso_optimization(n_particles, n_hidden_layers, n_nodes, iterations, X_train, y_train):
    particles = [Particle(n_hidden_layers, n_nodes) for _ in range(n_particles)]
    global_best_position = None
    global_best_error = float('inf')

    for _ in range(iterations):
        for particle in particles:
            error = train_mlp(particle.position, X_train, y_train)
            if error < particle.best_error:
                particle.best_error = error
                particle.best_position = particle.position.copy()
            if error < global_best_error:
                global_best_error = error
                global_best_position = particle.position.copy()

            particle.velocity = particle.velocity + np.random.rand() * (particle.best_position - particle.position)
            particle.position += particle.velocity

    return global_best_position

# Hyperparameters for PSO
n_particles = 200
n_hidden_layers = 30  # Changeable
n_nodes = 10          # Changeable
iterations = 500
```

```
# Train an initial model to create training data for PSO optimization
X_train = X
y_train = y

best_hyperparams = pso_optimization(n_particles, n_hidden_layers, n_nodes, iterations, X_train, y_train)

# Perform cross-validation to calculate MAE for each fold and predictions
mae_per_fold, all_predictions = cross_validate(X, y, n_folds=10)

# Print MAE for each fold
for fold, mae in enumerate(mae_per_fold, start=1):
    print(f'MAE for Fold {fold}: {mae}')

# Calculate and print overall MAE
overall_mae = np.mean(mae_per_fold)
print(f'Overall MAE across folds: {overall_mae}')

# Plotting MAE for each fold
folds = np.arange(1, len(mae_per_fold) + 1)

plt.figure(figsize=(12, 5))
plt.plot(folds, mae_per_fold, marker='o', color='b', linestyle='-', linewidth=2, markersize=8)
plt.title('Mean Absolute Error (MAE) for Each Fold')
plt.xlabel('Fold Number')
plt.ylabel('MAE')
plt.xticks(folds)

plt.grid(True)
plt.tight_layout()
plt.show()

# Plotting actual vs predicted values as a bar graph
plt.figure(figsize=(12, 6))
```

```
indices = np.arange(len(y))
bar_width = 0.35

plt.bar(indices, y, width=bar_width, label='Actual', color='b', alpha=0.6)
plt.bar(indices + bar_width, all_predictions, width=bar_width, label='Predicted', color='r', alpha=0.6)

plt.title('Actual vs Predicted Benzene Concentration')
plt.xlabel('Samples')
plt.ylabel('Benzene Concentration')
plt.legend()
plt.grid(axis='y')
plt.tight_layout()
plt.show()
```