รายงานการทดลอง

Computer Assignment 1

นายธนวัตน์ บำเพ็งพันธุ์ 630610736

CPE 261456

(Introduction to Computational Intelligence)

ภาคเรียนที่ 1 ปีการศึกษา 2565

การทดลองที่ 1
การ Predict ระดับน้ำที่สะพานนวรัตน์

ทำการทดลองการ Predict ระดับน้ำที่สะพานนวรัตน์ ในอีก 7 ชั่วโมงข้างหน้า โดยใช้ข้อมูลที่สถานี 1 และ สถานี 2 ที่เวลาปัจจุบัน และย้อนหลังไป 3 ชั่วโมง จากไฟล์ Flood dataset มีลักษณะของข้อมูลคือ สถานี 1 เวลา t – 3, สถานี 1 เวลา t – 2, สถานี 1 เวลา t – 1, สถานี 1 เวลา t - 0, สถานี 2 เวลา t – 3, สถานี 2 เวลา t – 2, สถานี 2 เวลา t – 1, สถานี 2 เวลา t – 0, Desire Output

1.1 วิธีการทดลอง

1.1.1 สร้าง Multi-layer Perceptron ที่มีรูปแบบดังนี้

Model				Activation function	
No.	Layer	Learning rate (LR)	Momentum rate (MR)	Hidden layer	Output layer
1.	8-4-1	0.01	0.01	Sigmoid	Linear
2.	8-4-1	0.05	0.03	Sigmoid	Linear
3.	8-8-1	0.02	0.01	Sigmoid	Linear
4.	8-2-2-1	0.01	0.01	Sigmoid	Linear

ตารางที่ 1: รายละเอียดของ Multi-layer Perceptron แต่ละตัวที่ใช้ในการทดลองที่ 1

Multi-layer Perceptron แต่ละตัวใช้ค่า Bias เท่ากับ 1.0 ในแต่ละ Neurons และ Weights ที่สุ่มในตอนเริ่มต้นจะอยู่ในช่วง [-1,1] และทุก Models มี Max epoch ที่ 2000 epoch

สมการของ Activation function $Sigmoid(x) = \frac{1}{1+e^{-x}}$, $Linear(x) = x$

1.1.2 ทำการ Standardization ข้อมูลใน Flood dataset

โดยข้อมูลใหม่ที่ได้จะมาจากสูตร $y = \frac{x - mean}{\text{standard deviation}}$ โดย x คือข้อมูลเดิมแต่ละค่าใน Flood dataset

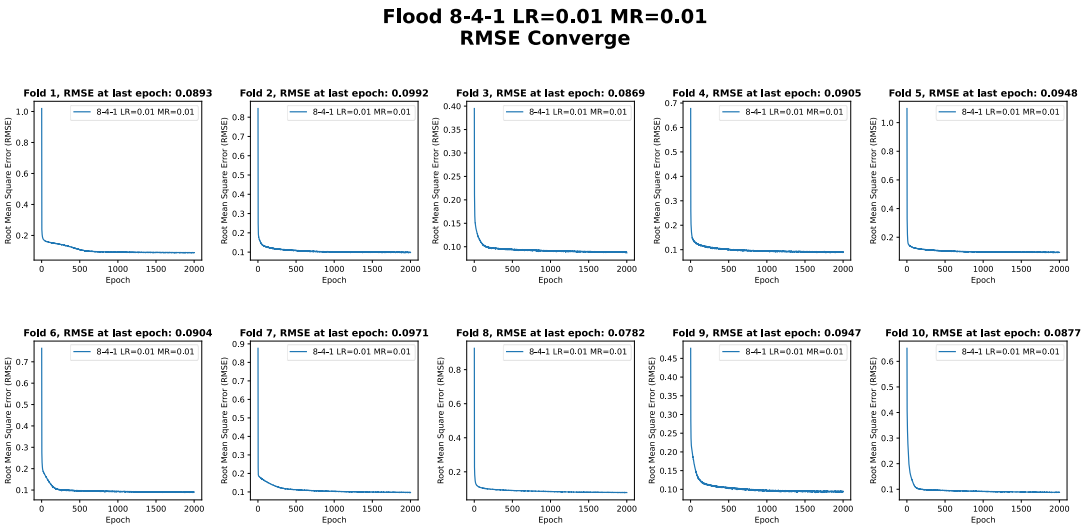
1.1.3 ทำ 10-fold Cross-validation ในแต่ละ Model จากตารางที่ 1

- ทำการสุ่มสลับที่ข้อมูลใน Flood dataset ที่ standardization แล้ว
- แบ่งชุดข้อมูล Flood dataset ที่ได้ออกเป็น 10 ชุดข้อมูลย่อยโดยที่ในแต่ละชุดข้อมูลมีจำนวนข้อมูลประมาณเท่าๆกัน และทำการ Training Model ด้วยชุดข้อมูล 9 ชุดจนครบ Max epoch และใช้ชุดข้อมูลย่อยที่เหลือในการ Validation Model และคำนวณ Root mean square error (RMSE) หลังจาก Training และ Validation เสร็จแล้ว ทำกระบวนการนี้ซ้ำ 10 ครั้งด้วยชุดข้อมูลย่อยสำหรับการ Validation ที่ไม่ซ้ำกัน
- ทำการบันทึก Error ของ Multi-layer Perceptron แต่ละ Model ทั้งระหว่างการ Training และ Validation ในแต่ละครั้ง Cross-validation

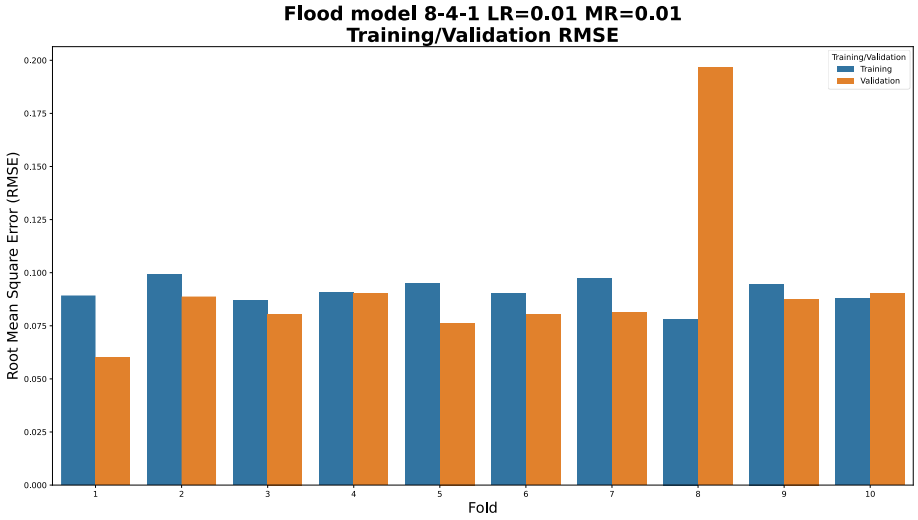
1.2 ผลการทดลอง

ผลลัพธ์ที่ได้จากการทำ 10-fold Cross-validation บน Flood dataset ในแต่ละ Models ในตารางที่ 1

1.2.1 Model 1: 8-4-1, Learning rate (LR) = 0.01, Momentum rate (MR) = 0.01

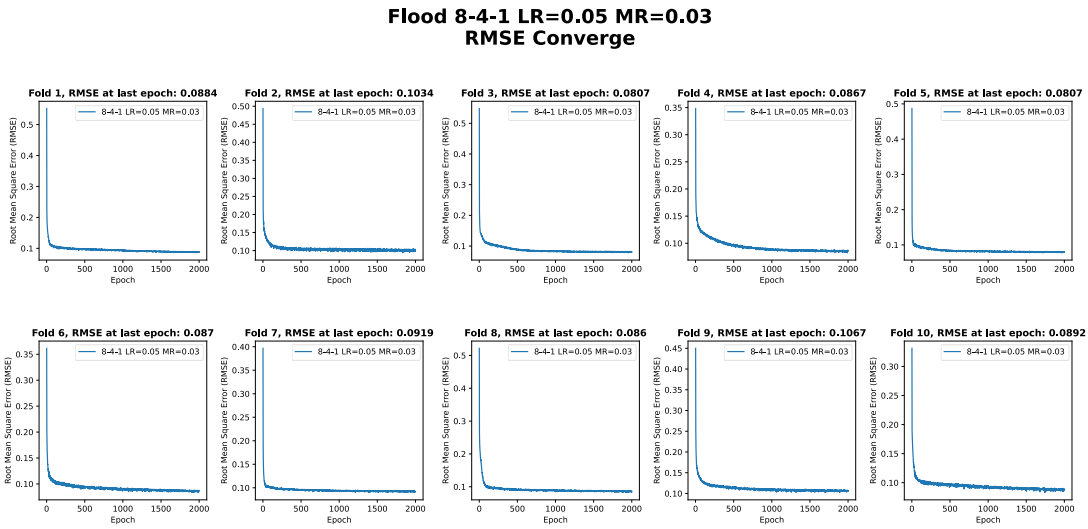


รูปที่ 1: 8-4-1, LR=0.01, MR=0.01 กราฟแสดงการลู่เข้าของ RMSE บน Training set ในแต่ละ Fold

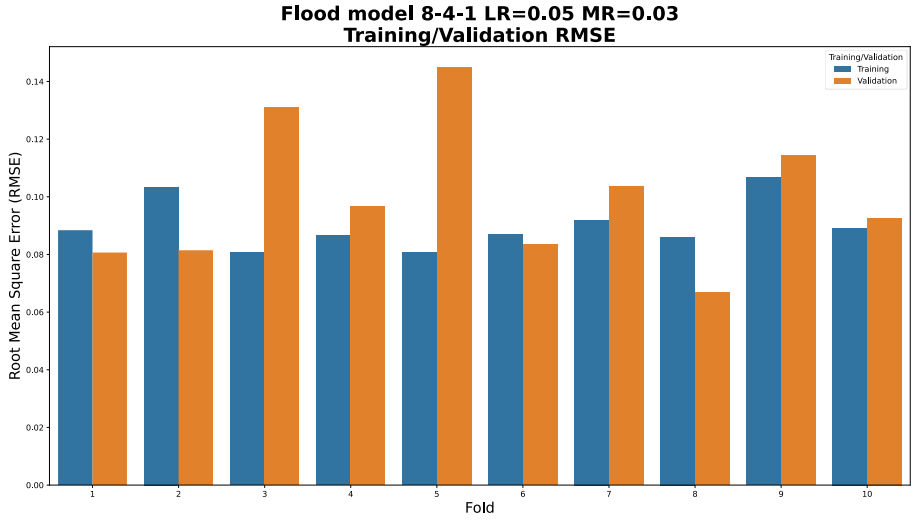


รูปที่ 2: 8-4-1, LR=0.01, MR=0.01 กราฟแสดงการเปรียบเทียบ RMSE ระหว่าง Training set (สีฟ้า) และ Validation set (สีส้ม) ที่ Max epoch ในแต่ละ Fold

1.2.2 Model 2: 8-4-1, Learning rate (LR) = 0.05, Momentum rate (MR) = 0.03

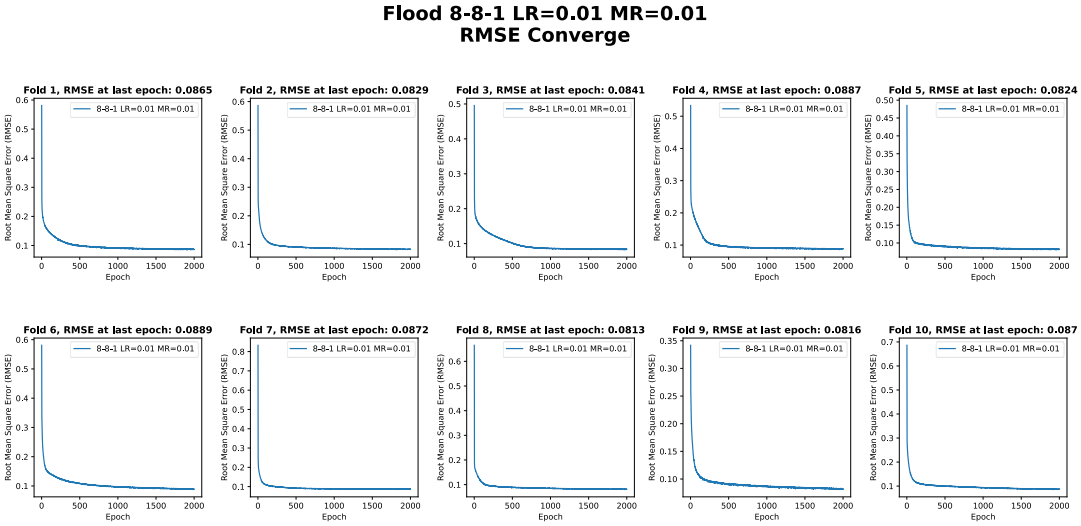


รูปที่ 3: 8-4-1, LR=0.05, MR=0.03 กราฟแสดงการลู่เข้าของ RMSE บน Training set ในแต่ละ Fold

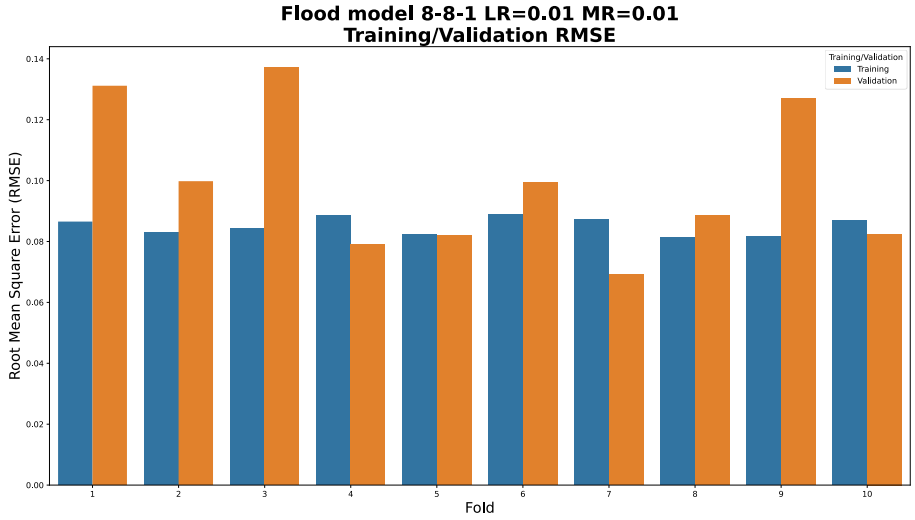


รูปที่ 4: 8-4-1, LR=0.05, MR=0.03 กราฟแสดงการเปรียบเทียบ RMSE ระหว่าง Training set (สีฟ้า) และ Validation set (สีส้ม) ที่ Max epoch ในแต่ละ Fold

1.2.3 Model 3: 8-8-1, Learning rate (LR) = 0.01, Momentum rate (MR) = 0.01

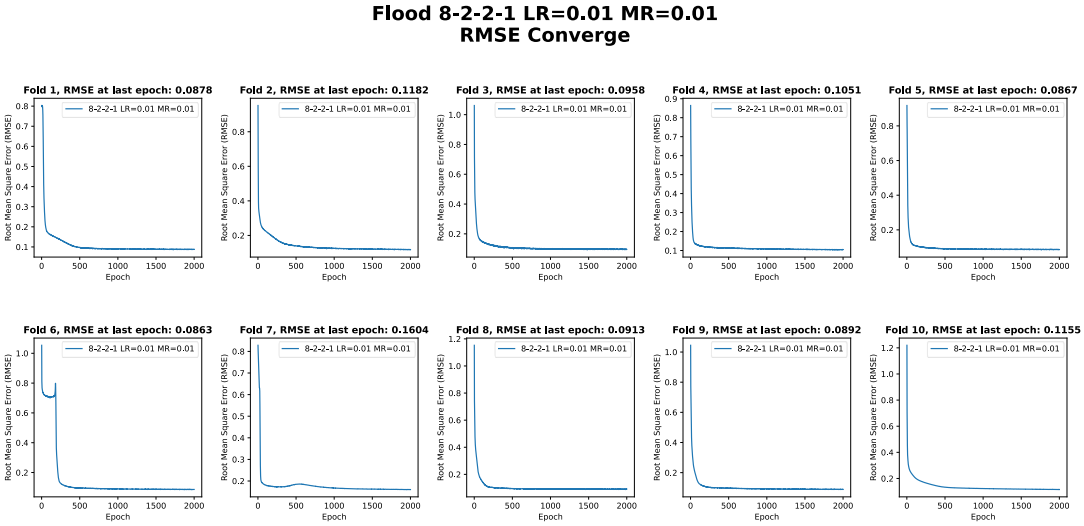


รูปที่ 5: 8-8-1, LR=0.01, MR=0.01 กราฟแสดงการลู่เข้าของ RMSE บน Training set ในแต่ละ Fold

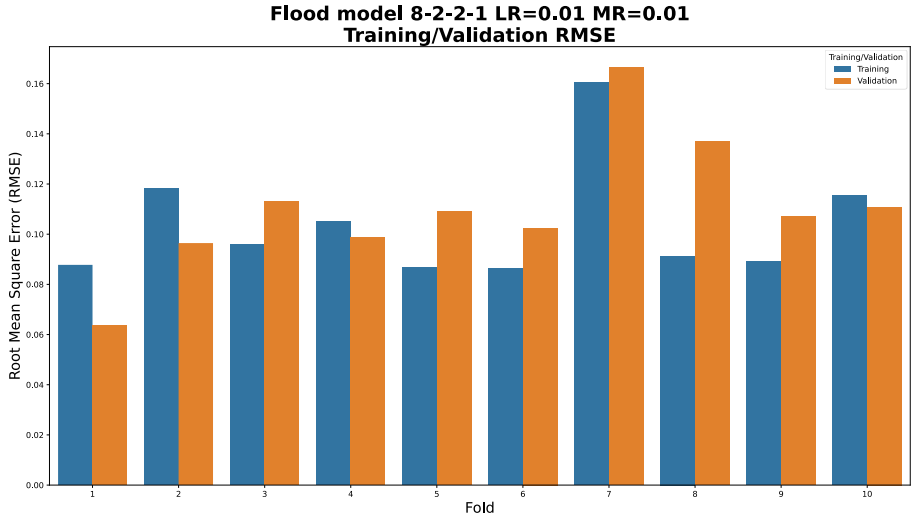


รูปที่ 6: 8-8-1, LR=0.01, MR=0.01 กราฟแสดงการเปรียบเทียบ RMSE ระหว่าง Training set (สีฟ้า) และ Validation set (สีส้ม) ที่ Max epoch ในแต่ละ Fold

1.2.4 Model 4: 8-2-2-1, Learning rate (LR) = 0.01, Momentum rate (MR) = 0.01



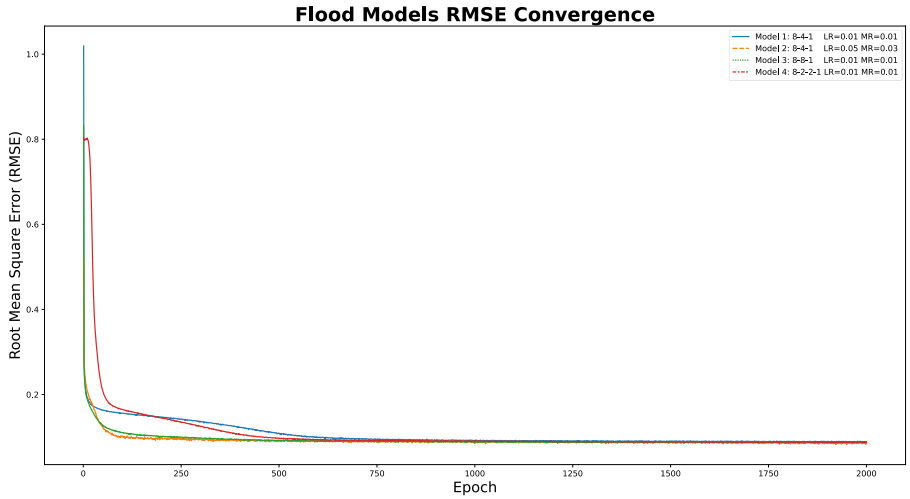
รูปที่ 7: 8-2-2-1, LR=0.01, MR=0.01 กราฟแสดงการลู่เข้าของ RMSE บน Training set ในแต่ละ Fold



รูปที่ 8: 8-2-2-1, LR=0.01, MR=0.01 กราฟแสดงการเปรียบเทียบ RMSE ระหว่าง Training set (สีฟ้า) และ Validation set (สีส้ม) ที่ Max epoch ในแต่ละ Fold

1.3 วิเคราะห์ผลการทดลอง

จากการทดลองที่ 1 ได้มีการสร้าง Model ขึ้นมาทดลองกับปัญหาทั้งหมด 4 ตัว โดยสรุปได้ผลลัพธ์ดังรูปและตารางนี้



รูปที่ 9: กราฟแสดงเปรียบเทียบการลู่เข้าของ RMSE ระหว่าง Training จาก Fold ที่มี RMSE บน Validation set ต่ำสุด ของทั้ง 4 model

Model no.	ค่าเฉลี่ยของ RMSE บน Validation set จากทั้ง 10 fold
1.	0.093235
2.	0.099520
3.	0.099567
4.	0.110448

ตารางที่ 2: แสดงค่าเฉลี่ย RMSE บน Validation set ของทั้ง 4 model

จากรูปที่ 9 เมื่อลองมาดูความเร็วในการลู่เข้าของ RMSE ในระหว่างการ Training ของแต่ละ Model จะเห็นว่าเมื่อเปรียบเทียบระหว่าง Model 1 (เส้นสีฟ้า) และ Model 2 (เส้นสีส้ม) ที่มี Layer เหมือนกันความเร็วในการลู่เข้าของ RMSE ในช่วงแรก Model 2 ลู่เข้าเร็วกว่าเนื่องมาจาก Learning rate ที่มากกว่า แต่จะเห็นว่าเส้นกราฟของ Model 2 นั้นมีการกวัดแกว่งของ RMSE สูงกว่า Model 1, Model 3 (เส้นสีเขียว) ที่ปรับจำนวน Neurons ใน Hidden layer เป็น 8 เมื่อเทียบกับ Model 1 มีการลู่เข้าเร็วกว่า Model 1 และใกล้เคียงกับ Model 2 ในช่วงแรก, Model 4 (เส้นสีแดง) มีความเร็วในการลู่เข้าใกล้เคียงกับ Model 1 และในช่วงรอบท้ายของการ Training ทั้ง 4 มี RMSE ใกล้เคียงกัน

จากตารางที่ 2 จะเห็นว่า Model 1 มีค่าเฉลี่ย Root mean square error (RMSE) บน Validation set ต่ำที่สุด, Model 2 ซึ่งมี Layer เหมือนกันกับ Model 1 แต่ทำการปรับเปลี่ยน Learning rate, Momentum rate และส่งผลกับ RMSE บน Validation set อย่างเห็นได้ชัด, Model 3 ได้ทำการปรับจำนวน Neurons ใน Hidden layer เป็น 8 ได้ RMSE เฉลี่ยใกล้เคียงกับ Model 2, Model 4 ได้ลองลดจำนวน Neurons ใน Hidden layer ลง แต่เพิ่ม Hidden layer เป็น 2 ชั้น ซึ่งเป็น Model ที่มี RMSE เฉลี่ยสูงสุดจากทั้ง 4 model

จึงสรุปการทดลองที่ 1 ได้ว่า Model 1 ให้ผลลัพธ์ที่ดีที่สุดจากทั้ง 4 Model ซึ่งมีรูปแบบดังนี้ Hidden layer 1 ชั้นและมี 4 Neurons, Learning rate และ Momentum rate อยู่ที่ 0.01 แล้วการเพิ่มจำนวน Neurons และชั้น Hidden layer ไม่ได้ทำให้ผลลัพธ์ดีขึ้นแต่แตกต่างกันในเรื่องของความเร็วในการลู่เข้าของ RMSE

การทดลองที่ 2

การจำแนกประเภท Class ของ Cross

การทดลองการจำแนกประเภท Class โดยใช้ข้อมูลจากไฟล์ Cross.pat ซึ่งมีลักษณะข้อมูลที่มี Input เป็นเลขทศนิยม 2 Input มาจำแนกประเภท Class 2 ประเภท และมี Desire Output 2 Output ที่ใช้บอก Class

2.1 วิธีการทดลอง

2.1.1 สร้าง Multi-layer Perceptron ที่มีรูปแบบดังนี้

Model				Activation function	
No.	Layer	Learning rate (LR)	Momentum rate (MR)	Hidden layer	Output layer
1.	2-4-2	0.01	0.01	ReLU	Linear
2.	2-4-2	0.005	0.06	ReLU	Linear
3.	2-8-2	0.01	0.01	ReLU	Linear
4.	2-4-4-2	0.01	0.01	ReLU	Linear

ตารางที่ 3: รายละเอียดของ Multi-layer Perceptron แต่ละตัวที่ใช้ในการทดลองที่ 2

Multi-layer Perceptron แต่ละตัวใช้ค่า Bias เท่ากับ 1.0 ในแต่ละ Neurons และ Weights ที่สุ่มในตอนเริ่มต้นจะอยู่ในช่วง [-1,1] และทุก Models มี Max epoch ที่ 2500 epoch

สมการของ Activation function $\mathrm{Re}\,LU(x) = \max(0, x)$, $Linear(x) = x$

2.1.2 ทำ 10-fold Cross-validation ในแต่ละ Model จากตารางที่ 1

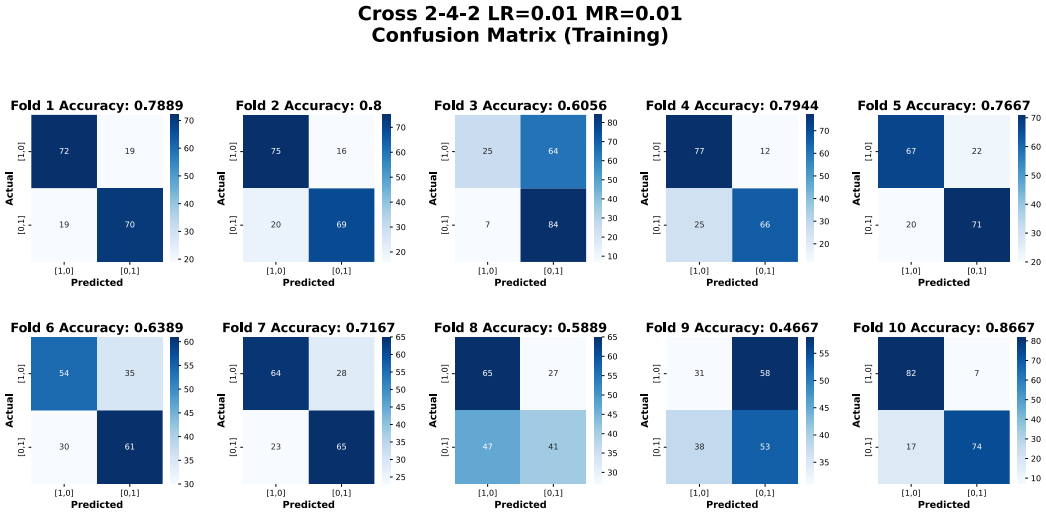
- แบ่งชุดข้อมูลในไฟล์ Cross.pat แบบสุ่มออกเป็น 10 ชุดข้อมูลย่อยโดยที่ในแต่ละชุดข้อมูลมีจำนวนข้อมูลประมาณเท่าๆกัน และทำการ Training Model ด้วยชุดข้อมูล 9 ชุดจนครบ Max epoch และใช้ชุดข้อมูลย่อยที่เหลือในการ Validation Model และคำนวณ Predicted output กับ Actual output ไว้ใน Confusion Matrix หลังจาก Training และ Validation เสร็จแล้ว ทำกระบวนการนี้ซ้ำ 10 ครั้งด้วยชุดข้อมูลย่อยสำหรับการ Validation ที่ไม่ซ้ำกัน

- ทำการบันทึก Confusion Matrix และหา Accuracy โดยที่ $Accuracy = \frac{\sum TP + TN}{\sum TP + FP + FN + TN}$ ของ Multi-layer Perceptron แต่ละ Model ทั้งระหว่างการ Training และ Validation ในแต่ละครั้ง Cross-validation

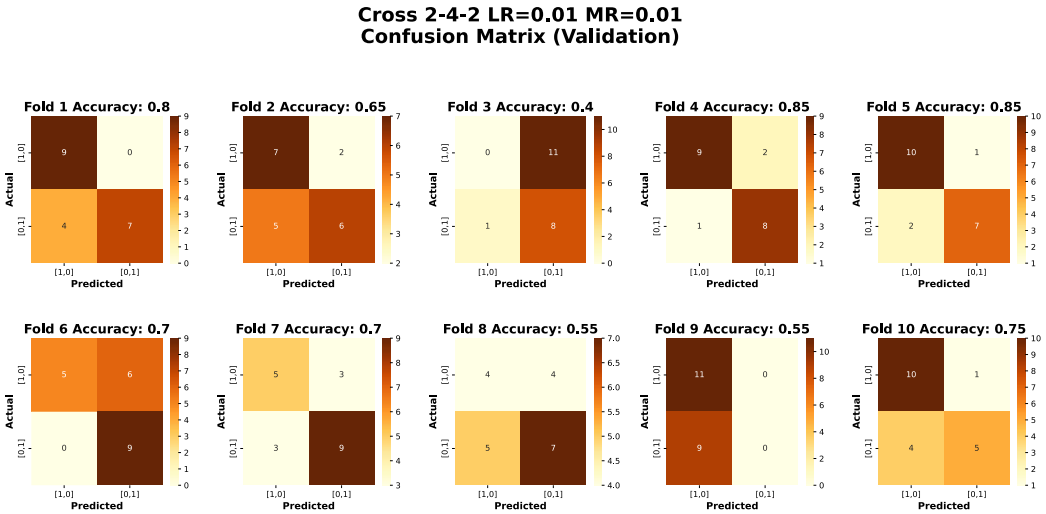
2.2 ผลการทดลอง

ผลลัพธ์ที่ได้จากการทำ 10-fold Cross-validation บนไฟล์ Cross.pat ในแต่ละ Model ในตารางที่ 3

2.2.1 Model 1: 2-4-2, Learning rate (LR) = 0.01, Momentum rate (MR) = 0.01

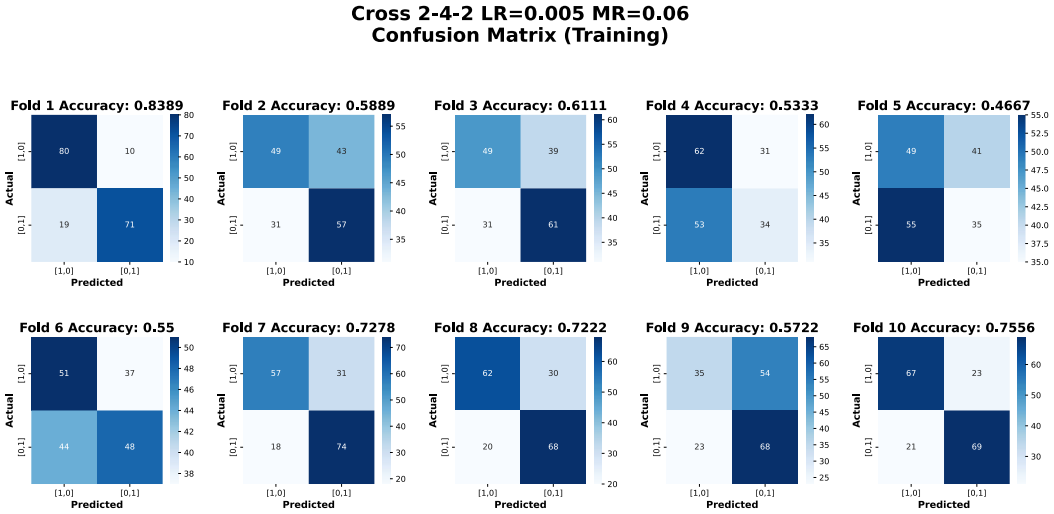


รูปที่ 10: 2-4-2, LR=0.01, MR=0.01 กราฟแสดง Confusion Matrix ของ Training set ในแต่ละ Fold

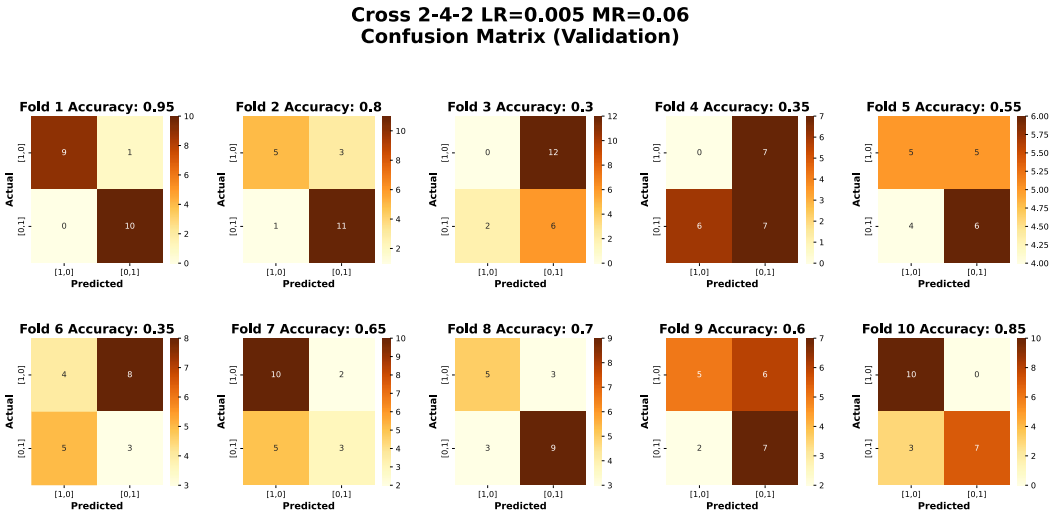


รูปที่ 11: 2-4-2, LR=0.01, MR=0.01 กราฟแสดง Confusion Matrix ของ Validation set ในแต่ละ Fold

2.2.2 Model 2: 2-4-2, Learning rate (LR) = 0.005, Momentum rate (MR) = 0.06

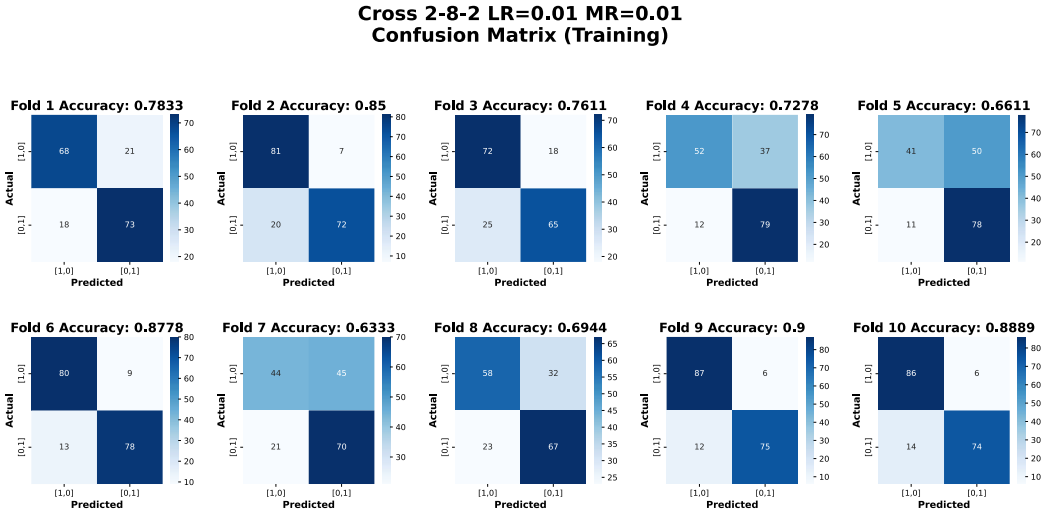


รูปที่ 12: 2-4-2, LR=0.005, MR=0.06 กราฟแสดง Confusion Matrix ของ Training set ในแต่ละ Fold

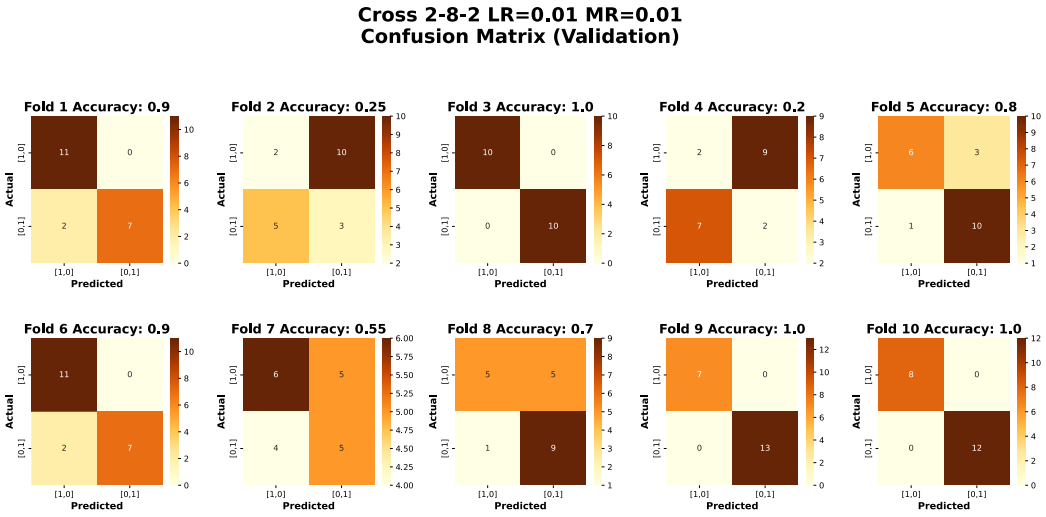


รูปที่ 13: 2-4-2, LR=0.005, MR=0.06 กราฟแสดง Confusion Matrix ของ Validation set ในแต่ละ Fold

2.2.3 Model 3: 2-8-2, Learning rate (LR) = 0.01, Momentum rate (MR) = 0.01

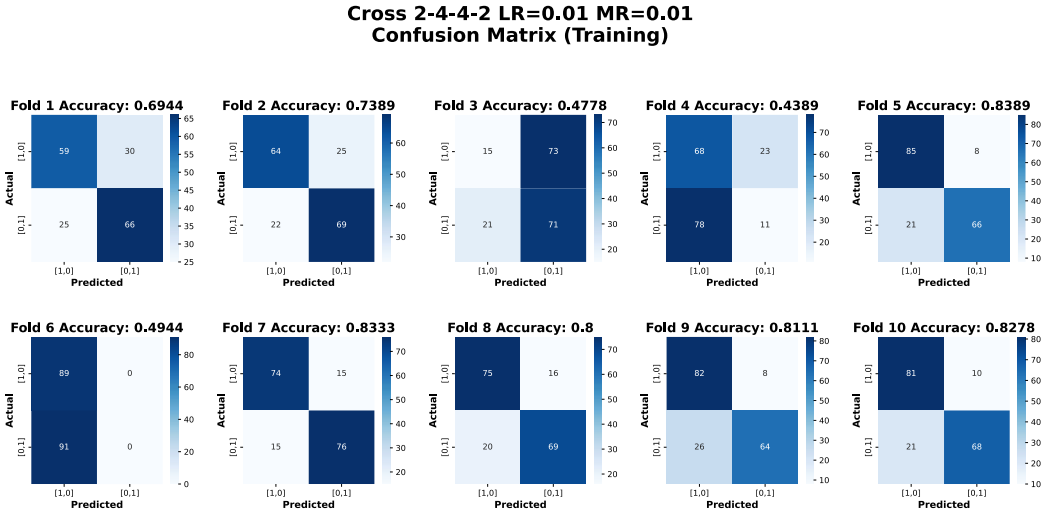


รูปที่ 14: 2-8-2, LR=0.01, MR=0.01 กราฟแสดง Confusion Matrix ของ Training set ในแต่ละ Fold

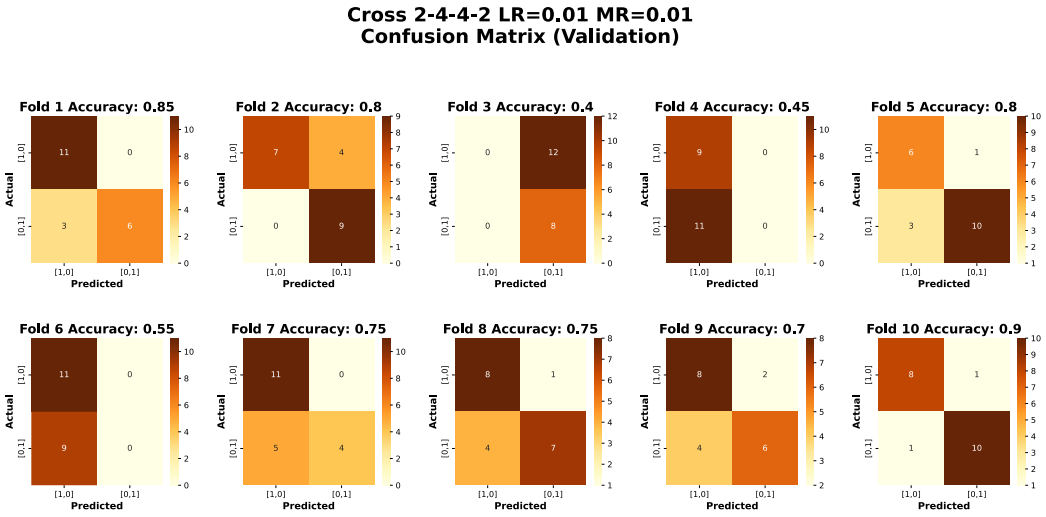


รูปที่ 15: 2-8-2, LR=0.01, MR=0.01 กราฟแสดง Confusion Matrix ของ Validation set ในแต่ละ Fold

2.2.4 Model 4: 2-4-4-2, Learning rate (LR) = 0.01, Momentum rate (MR) = 0.01



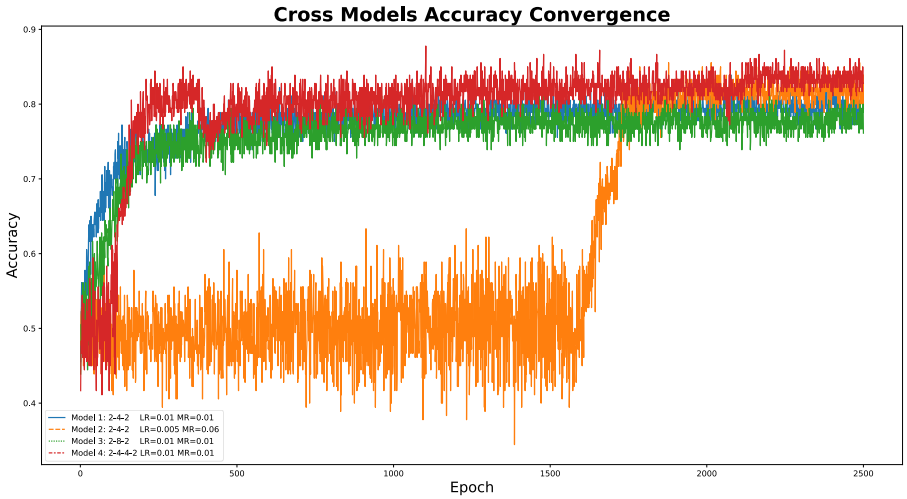
รูปที่ 16: 2-4-4-2, LR=0.01, MR=0.01 กราฟแสดง Confusion Matrix ของ Training set ในแต่ละ Fold



รูปที่ 17: 2-4-4-2 LR=0.01, MR=0.01 กราฟแสดง Confusion Matrix ของ Validation set ในแต่ละ Fold

2.3 วิเคราะห์ผลการทดลอง

จากการทดลองที่ 2 ได้มีการสร้าง Model ขึ้นมาทดลองกับปัญหาทั้งหมด 4 ตัว โดยสรุปได้ผลลัพธ์ดังรูปและตารางนี้



รูปที่ 17: กราฟแสดงเปรียบเทียบการลู่เข้าของ Accuracy ระหว่าง Training จาก Fold ที่มี Accuracy บน Validation set สูงที่สุด ของทั้ง 4 model

Model No.	ค่าเฉลี่ย Accuracy บน Training set จากทั้ง 10 fold	ค่าเฉลี่ย Accuracy บน Validation set จากทั้ง 10 fold
1.	70.335%	68.000%
2.	63.667%	61.000%
3.	77.777%	73.000%
4.	69.555%	69.500%

ตารางที่ 4: แสดงค่าเฉลี่ย Accuracy บน Training/Validation set ของทั้ง 4 model

จากรูปที่ 17 จะเห็นว่า Model 1 (เส้นสีฟ้า), Model 3 (เส้นสีเขียว), Model 4 (เส้นสีแดง) มีความเร็วในการลู่เข้าของ Accuracy พอๆกันในช่วงแรก แต่จะเห็นความแตกต่างอย่างชัดเจนกับ Model 2 (เส้นสีส้ม) ซึ่งมี Layer เหมือนกับ Model 1 แตกต่างกันที่ Learning rate, Momentum rate มีความเร็วในการลู่เข้าของ Accuracy ที่ช้ากว่าและกวัดแกว่งกว่ามากๆ แต่เมื่อมาเปรียบเทียบในช่วงรอบท้ายของการ Training Model 2 มี Accuracy ที่สูงใกล้เคียงกับ Model 4 และ Model 1 ใกล้เคียงกับ Model 3

จากตารางที่ 4 เมื่อเปรียบเทียบ Accuracy ของ Model 1 กับ Model 2 ที่มีการปรับค่า Learning rate, Momentum rate จะเห็นว่าส่งผลกับ Accuracy อย่างชัดเจนทั้ง Training set และ Validation set และเมื่อลองปรับ Model 1 โดยการเพิ่มจำนวน Neurons ใน Hidden layer จาก 4 เป็น 8 ซึ่งก็คือ Model 3 จะเห็นว่าค่าเฉลี่ย Accuracy เพิ่มขึ้นอย่างเห็นได้ชัดทั้ง Training set และ Validation set และการแบ่ง Hidden layer ออกเป็น 2 ชั้นดัง Model 4 ได้ค่าเฉลี่ยใกล้เคียงกับ Model 1

จึงสรุปการทดลองที่ 2 ได้ว่า Model 3 ให้ค่าเฉลี่ย Accuracy สูงที่สุดจากทั้ง 4 Model โดยมีรูปแบบดังนี้ Hidden layer 1 ชั้นและมี 8 Neuron ค่า Learning rate, Momentum rate อยู่ที่ 0.01 เมื่อเทียบกับ Model 1 จะเห็นว่าการเพิ่มจำนวน Neurons ในชั้น Hidden layer บนปัญหานี้ทำให้ได้ผลลัพธ์ที่ดีขึ้น

สรุปผลจากทั้ง 2 การทดลอง

จากการทดลองที่ 1 จะเห็นว่าเมื่อเราลองเพิ่มจำนวน Neurons ในชั้น Hidden layer กลับไม่ได้ทำให้ได้ผลลัพธ์ดีขึ้นซึ่งแตกต่างกับผลลัพธ์ในการทดลองที่ 2 ดังนั้นรูปแบบของ Multi-layer Perceptron นั้นขึ้นอยู่กับปัญหา ไม่มีรูปแบบที่ตายตัวที่ทำงานได้ดีกับทุกๆปัญหา เราต้องลองปรับเปลี่ยน Layer และจำนวน Neurons ต่างๆรวมถึงค่า Learning rate, Momentum rate หากมีการปรับค่าที่ดีและเหมาะสมกับปัญหาก็จะทำให้ Model นั้นๆให้ผลลัพธ์ที่ดีขึ้นด้วย