ระบบตรวจสอบคุณภาพก้อนเพาะเชื้อเห็ดนางฟ้าภูฏาน

QUALITY INSPECTION OF MUSHROOM SPAWN BY VISUAL ANALYSIS

พิชญา สุภา

พงศกร ทองแช่ม

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมระบบควบคุม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2562

QUALITY INSPECTION OF MUSHROOM SPAWN BY VISUAL ANALYSIS

PICHAYA SUPA

PHONGSAKHON TONGCHAM

THIS THESIS IS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENTS FOR THE DEGREE OF

BACHELOR OF ENGINEERING IN CONTROL ENGINEERING

FACULTY OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

ACADEMIC YEAR 2019

ระบบตรวจสอบคุณภาพก้อนเพาะเชื้อเห็ดนางฟ้าภูฏาน

โดย
นายพงศกร ทองแซ่ม 59010885
นางสาวพิชญา สุภา 59010965

อาจารย์ที่ปรึกษา
ผู้ช่วยศาสตราจารย์ ดร.พิชชา ประสิทธิ์มีบุญ

ปีการศึกษา 2562

บทคัดย่อ

งานวิจัยนี้นำเสนอรายละเอียดในการสร้างแบบจำลอง ที่เกิดจากการเรียนรู้ของเครื่องเพื่อประยุกต์ใช้ในการคัดแยกคุณภาพก้อนเพาะเชื้อเห็ดนางฟ้าภูฏาน โดยมีวัตถุประสงค์เพื่อพัฒนาระบบการคัดแยกให้มีความสะดวก รวดเร็วและแม่นยำมากขึ้น เนื่องจากเดิมเกษตรกรไทยจะอาศัยแรงงานมนุษย์ในการคัดแยก ซึ่งทำให้เกิดความเมื่อยล้าและความผิดพลาดของมนุษย์ได้ ดังนั้นคณะผู้จัดทำจึงได้ศึกษาวิธีการประมวลภาพเพื่อการคัดแยกคุณภาพก้อนเพาะเชื้อเห็ดนางฟ้า โดยอาศัยข้อมูลจากฮีสโตแกรมของก้อนเพาะเชื้อเห็ดนางฟ้าภูฏาน นำข้อมูลมาผ่านกระบวนการก่อนการประมวลผล (Pre-processing) เพื่อคัดกรองข้อมูลที่ต้องการ และนำมาคัดแยกโดยใช้วิธีของการเรียนรู้ของเครื่อง (Machine Learning) 4 วิธี เพื่อแบ่งคุณภาพก้อนเพาะเชื้อเห็ดนางฟ้าภูฏานออกเป็น 3 กลุ่ม โดยแบบจำลองที่นำเสนอสามารถระบุและคัดแยกคุณภาพก้อนเพาะเชื้อเห็ดได้อย่างถูกต้องและแม่นยำ

QUALITY INSPECTION OF MUSHROOM SPAWN BY VISUAL ANALYSIS

By

Mr. Phongsakhon Tongcham 59010885

Miss Pichaya Supa 59010965

Advisor

Asst.Prof.Dr. Pitcha Prasitmeeboon

Academic Year 2019

ABSTRACT

This research presents a model based on machine learning methods for quality inspection of mushroom spawns. Thai famers normally rely on human labors to inspect mushroom spawn quality which can cause human errors. Therefore, we studied the image processing methods to create histograms of mushroom spawns images and apply pre-processing techniques to obtain useful data. Using 4 methods in machine learning could classify spawns into 3 main classes. The proposed model could identify and classify the quality of mushroom spawn accurately.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้ประสบความสำเร็จลุล่วงไปได้ด้วยดี อันเนื่องจากความกรุณาของอาจารย์ที่ปรึกษา ผู้ช่วยศาสตราจารย์ ดร.พิชชา ประสิทธิ์มีบุญ ที่ได้ให้ความรู้ คำปรึกษา ชี้แนะแนวทางในการแก้ปัญหา และประสบการณ์ที่ดีแก่ข้าพเจ้า สนับสนุนอุปกรณ์ที่จำเป็นตลอดจนตรวจแก้ไขข้อบกพร่องต่างๆ ด้วยความเอาใจใส่เป็นอย่างยิ่งตลอดระยะเวลาที่ได้ทำการศึกษาจัดทำปริญญานิพนธ์นี้

ขอขอบคุณผู้แต่งหนังสือ เอกสารอ้างอิง และเว็บไซต์ต่างๆ ที่คณะผู้จัดทำได้นำมาใช้อ้างอิงประกอบการศึกษา และจัดทำปริญญานิพนธ์ฉบับนี้จนปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงไปด้วยดี

ขอขอบคุณครอบครัวอันเป็นที่รัก ครูอาจารย์ที่เคารพ ตลอดจนเพื่อนนักศึกษาทุกท่านที่คอยถามไถ่ความคืบหน้า ให้คำแนะนำและให้กำลังใจที่ดีเสมอมา

สำหรับคุณงามความดีอันใดที่เกิดจากปริญญานิพนธ์ฉบับนี้ ข้าพเจ้าขอมอบให้กับบิดามารดา ซึ่งเป็นที่รักและเคารพยิ่ง ตลอดจนครูอาจารย์ที่เคารพทุกท่านที่ได้ประสิทธิ์ประสาทวิชาความรู้และถ่ายทอดประสบการณ์ที่ดีให้แก่ข้าพเจ้า

คณะผู้จัดทำ

พงศกร ทองแช่ม

พิชญา สุภา

สารบัญ

	หน้า
บทคัดย่อภาษาไทย	**I**
บทคัดย่อภาษาอังกฤษ	**II**
กิตติกรรมประกาศ	**III**
สารบัญ	**IV**
สารบัญรูป	**VII**
สารบัญตาราง	**IX**
บทที่ 1 บทนำ	**1**
1.1 ที่มาและความสำคัญ	1
1.2 วัตถุประสงค์	1
1.3 ขอบเขตโครงงาน	1
บทที่ 2 ทฤษฎีที่เกี่ยวข้อง	**2**
2.1 เห็ดนางฟ้าภูฏาน	2
2.1.1 ลักษณะทั่วไป	2
2.1.2 การเพาะเห็ดนางฟ้าภูฏาน	3
2.2 โครงข่ายประสาทเทียม (Artificial Neural Networks)	4
2.2.1 เซลล์ประสาท (Neurons)	5
2.2.2 การออกแบบ Artificial Neural Network	7
2.2.3 การเรียนรู้ของ Machine Learning	8
2.2.4 ฟังก์ชันการกระตุ้น (Activation Function)	9
2.3 K–Nearest Neighbors (KNN)	14
2.3.1 หลักการทำงานของขั้นตอนวิธี K-Nearest Neighbors	14
2.3.2 ตัวอย่างการทำงานของขั้นตอนวิธี K-Nearest Neighbors	14
2.3.3 ข้อควรระวังในการใช้วิธี KNN	16

สารบัญ (ต่อ)

	หน้า
2.4 Nearest Centroid Classifier (NCC)	16
2.4.1 ตัวอย่างการทำงานของขั้นตอนวิธี Nearest Centroid	17
2.5 Support Vector Machine (SVM)	18
2.5.1 หลักการทำงานของของขั้นตอนวิธี Support Vector Machine	19
บทที่ 3 วิธีการทดลอง	**21**
3.1 วิธีการดำเนินงาน	21
3.2 ซอฟต์แวร์ที่เกี่ยวข้อง	21
3.3 ขั้นตอนการวางแผนและเก็บข้อมูล	23
3.4 การเตรียมข้อมูล	24
3.4.1 การดึงพื้นที่ที่สนใจ (Region of Interest) ออกจากพื้นหลัง	24
3.4.2 ขั้นตอนการประมวลผลภาพ (Pre-processing Step)	29
3.5 เริ่มต้นศึกษาวิธีการเขียนซอฟต์แวร์	29
3.6 วิธีการทำงานของโปรแกรม	30
3.7 การคัดแยกคุณภาพก้อนเพาะเชื้อเห็ดนางฟ้า	31
บทที่ 4 ผลการทดลอง	**32**
4.1 ผลการเก็บข้อมูล	32
4.2 ผลการวิเคราะห์	32
4.2.1 อัตราความแม่นยำจากการคัดแยกด้วยวิธี SVM และ NCC	32
4.2.2 อัตราความแม่นยำจากการคัดแยกด้วยวิธี KNN	33
4.2.3 อัตราความแม่นยำจากการคัดแยกด้วยวิธี ANN	33
บทที่ 5 สรุปผล	**34**
5.1 สรุปผลการดำเนินงาน	34
5.2 อุปสรรคในการดำเนินงาน	34
เอกสารอ้างอิง	**35**

สารบัญ (ต่อ)

หน้า

ภาคผนวก **39**

ภาคผนวก ก โปรแกรมการทำงานและการแสดงผล 40

สารบัญรูป

รูปที่	หน้า
2.1 เห็ดนางฟ้าภูฏาน	2
2.2 เซลล์ประสาทของมนุษย์	5
2.3 เพอร์เซปตรอน	6
2.4 โครงสร้างของ Artificial Neural Network	7
2.5 แนวคิดการทำงานของ Backpropagation	9
2.6 Linear Activation Function	10
2.7 Sigmoid Activation Function	11
2.8 Hypoberlic Tangent (Tanh) Activation Function	12
2.9 Rectified Linear Unit (ReLU) Activation Function	13
2.10 กลุ่มข้อมูลที่ใช้เป็น Training Data	14
2.11 มีข้อมูลใหม่ที่ต้องการพิจารณาเข้ามา	15
2.12 หาเพื่อนบ้านที่ใกล้ที่สุด 3 ตัว	15
2.13 จัดกลุ่มให้ข้อมูล	15
2.14 ปัญหาเมื่อ K เป็นเลขคู่สำหรับกลุ่มที่มีจำนวนคู่	16
2.15 ปัญหาเมื่อมีข้อมูล 3 กลุ่ม และ K=6 ซึ่งเป็นผลคูณของ 3	16
2.16 กลุ่มข้อมูล 3 กลุ่ม	17
2.17 มีข้อมูลใหม่ที่ต้องการพิจารณาเข้ามา	17
2.18 หา Centroid ของแต่ละกลุ่มข้อมูล	17
2.19 หา Nearest Centroid	18
2.20 จัดกลุ่มข้อมูล	18
2.21 ตัวอย่างการแบ่งข้อมูลด้วยวิธี SVM แบบ 2 มิติ	19
2.22 SVM แบบ Hard Margin และ Soft Margin	20
3.1 โปรแกรม Python	22
3.2 การเก็บรูปภาพก้อนเพาะเชื้อเห็ดนางฟ้า	23
3.3 ก้อนเพาะเชื้อเห็ดกลุ่ม A และ B	24
3.4 ก้อนเพาะเชื้อเห็ดกลุ่ม C1, C2, C3 และ C4 ตามลำดับ	24
3.5 ก้อนเพาะเชื้อเห็ดนางฟ้าภูฏาน	25
3.6 การสร้างเส้นขอบรูปภาพ	25
3.7 การ Close Gap	26

สารบัญรูป (ต่อ)

รูปที่	หน้า
3.8 การลบเส้นขอบที่ไม่ต้องการ	26
3.9 วาดเส้นวงที่พื้นที่มากที่สุดซึ่งล้อมรอบวัตถุ	27
3.10 ระบายสีขาวในพื้นที่เส้นวง	28
3.11 รูปภาพที่ทำการลบพื้นหลังเรียบร้อย	28
3.12 การ Cross Validation	30

สารบัญตาราง

ตารางที่	หน้า
4.1 อัตราความแม่นยำจากการประมวลผลด้วยวิธี SVM และ NCC	32
4.2 อัตราความแม่นยำจากการประมวลผลด้วยวิธี KNN	33
4.3 อัตราความแม่นยำจากการประมวลผลด้วยวิธี ANN	33

บทที่ 1

บทนำ

1.1 ที่มาและความสำคัญ

ในปัจจุบันประชากรไทยหันมาใส่ใจเรื่องสุขภาพและรับประทานผลผลิตจากธรรมชาติมากขึ้น ทั้งในชีวิตประจำวันและในช่วงเทศกาล เช่น เทศกาลกินเจ ส่งผลให้การเติบโตทางเศรษฐกิจทางด้านการเกษตรพืชผักผลไม้ของไทยเพิ่มขึ้นอย่างรวดเร็ว หนึ่งในนั้นคือ เห็ดนางฟ้าภูฏาน จัดเป็นเห็ดเศรษฐกิจที่เป็นที่นิยมบริโภคทั่วไป สามารถหาทานได้ตลอดปี อีกทั้งยังมีสรรพคุณทางยา และตลาดเห็ดยังมีการเติบโตมากขึ้นเรื่อยๆ จากการศึกษาข้อมูลเกี่ยวกับการเพาะเห็ดนางฟ้าภูฏานในประเทศไทย พบว่าแต่ละขั้นตอนอาศัยเวลาและความละเอียดในการควบคุมปัจจัยต่างๆ ในการเพาะเชื้อ และในขั้นตอนย้ายโรงเรือนเพื่อเปิดดอก จะอาศัยแรงงานคนในการตรวจสอบการเติบโตของเชื้อราในก้อนเพาะเชื้อเห็ด ซึ่งในขั้นตอนนี้อาจก่อให้เกิดความเมื่อยล้าและเกิดความผิดพลาดจากมนุษย์ ทั้งยังส่งผลกระทบกับความปลอดภัยด้านสุขภาพของเกษตรกรจากก๊าซคาร์บอนไดออกไซด์จำนวนมาก ที่เกิดขึ้นจากการเพาะเชื้อภายในโรงเรือน จึงเป็นแนวคิดให้จัดทำเครื่องตรวจสอบคุณภาพก้อนเพาะเชื้อเห็ดนางฟ้าภูฏาน เพื่อให้สามารถลด Human Errors ที่เกิดขึ้นในกระบวนการและเพิ่มความปลอดภัยด้านสุขภาพในการทำงานให้กับเหล่าเกษตรกร

1.2 วัตถุประสงค์

เพื่อพัฒนาแบบจำลองสำหรับการตรวจสอบก้อนเพาะเชื้อเห็ดนางฟ้าภูฏาน และลดความผิดพลาดที่เกิดขึ้นจากแรงงานมนุษย์

1.3 ขอบเขตโครงงาน

สร้างแบบจำลองทางคณิตศาสตร์ที่สามารถจำแนกก้อนเพาะเชื้อเห็ด ออกเป็นกลุ่มตามที่ต้องการได้อย่างเหมาะสมและแม่นยำ

บทที่ 2

ทฤษฎีที่เกี่ยวข้อง

2.1 เห็ดนางฟ้าภูฏาน

2.1.1 ลักษณะทั่วไป

เห็ดนางฟ้าภูฏาน (Indian Oyster Mushroom หรือ Phoenix Mushroom หรือ Lung Oyster) เป็นเห็ดนางรมชนิดหนึ่งซึ่งมีสายพันธุ์มาจากประเทศภูฏาน (มโนรัตน์ โทลักษณ์, 2010) จัดได้ว่าเป็นเห็ดเศรษฐกิจชนิดหนึ่งที่เป็นที่นิยมในการบริโภคเป็นเห็ดที่ออกดอกได้ดีในฤดูฝนและสามารถออกดอกสม่ำเสมอนาน 6-7 เดือน มีอายุการพักเชื้อที่สั้น เพาะง่าย สามารถเพาะได้ทุกฤดูแต่ไม่นิยมเพาะในฤดูร้อน เนื่องจากเห็ดชอบอากาศเย็นชื้น นอกจากจะมีการควบคุมความชื้นในโรงเรือนที่ดี โดยเห็ดนางฟ้าภูฏานจะเกิดเป็นกลุ่มที่มีโคนเชื่อมติดกัน กลุ่มละ 3-10 ดอก หมวกเห็ดเป็นรูปพัดหรือรูปใบพาย มีสีน้ำตาลอ่อนอมเทา เนื้อเห็ดสีขาว ใต้หมวกเห็ดจะมีครีบแคบเรียวยาวลงไปจนถึงก้านดอก ซึ่งจะไม่อยู่พอดีกับกึ่งกลางของดอก มักค่อนไปด้านใดด้านหนึ่ง มีความยาวประมาณ 1-4 เซนติเมตร ดังรูปที่ 2.1

รูปที่ 2.1 เห็ดนางฟ้าภูฏาน

ที่มา http://farmkaset.blogspot.com/2015/07/blog-post_82.html

2.1.2 การเพาะเห็ดนางฟ้าภูฏาน

เกษตรกรนิยมใช้ขี้เลื่อยไม้ยางพาราเป็นวัสดุเพาะ เนื่องจากให้ผลผลิตสูงและเก็บผลผลิตได้นาน (บุญส่ง วงศ์เกรียงไกร, 2000) มีวิธีการที่ซับซ้อนน้อยกว่าและใช้ระยะเวลาในการรอผลผลิตสั้นกว่าวัสดุอื่น

สูตรอาหารเพาะเห็ดนางฟ้าภูฏาน

1. ขี้เลื่อยไม้ยางพารา 100 กิโลกรัม

2. รำละเอียด 5 กิโลกรัม

3. ปูนขาว 1 กิโลกรัม

4. ยิปซั่ม 0.5 กิโลกรัม

5. ดีเกลือ 0.2 กิโลกรัม

6. ความชื้น 60-70 เปอร์เซ็นต์ (ดูตามสภาพของขี้เลื่อย)

วิธีการเพาะเชื้อ

1. ผสมอาหารเพาะเห็ดตามสูตร

2. บรรจุอาหารเพาะลงในถุงพลาสติกทนร้อน

3. รวบปากถุงบีบไล่อากาศออก แล้วอัดถุงให้แน่น

4. นำไปนึ่งฆ่าเชื้อด้วยถังนึ่ง ไม่อัดความดัน อุณหภูมิประมาณ 75-100 องศาเซลเซียส ประมาณ 3 ชั่วโมง

5. นำถุงอาหารเพาะเห็ดมาใส่เชื้อเห็ดนางฟ้าภูฏาน

6. นำมาพักเชื้อประมาณ 30 วัน โดยการเก็บไว้ในโรงพักเชื้อไม่ให้โดนแสงแดดและความชื้น หากพบว่าเกิดก้อนเชื้อราดำหรือราเขียว ให้รีบแยกออกเพื่อป้องกันการระบาดของเชื้อ

7. นำไปเปิดดอก โดยการเปิดจุกสำลีออก

8. ทำการเก็บผลผลิต โดยจับโคนเห็ดให้แน่นแล้วโยก ซ้าย-ขวา-บน-ล่าง แล้วดึงเห็ดออกมาทั้งโคน อย่าให้มีก้านดอกติดอยู่ในปากถุง ถ้ามีให้เขี่ยออกให้สะอาดเพราะจะทำให้ก้อนเพาะเชื้อเห็ดนางฟ้าเน่าได้

9. รดน้ำที่ผิวหน้าก้อนเพาะเชื้อเพื่อปรับสภาพความชื้น

ปัจจัยที่มีผลต่อการเจริญเติบโตของเห็ดนางฟ้าภูฏาน

1. อุณหภูมิ ต้องควบคุมให้เหมาะสมอยู่ที่ 22-28 องศาเซลเซียส

2. ความชื้นสัมพัทธ์ อยู่ที่ระหว่าง 70-90 เปอร์เซ็นต์

3. มีแสงแดดรำไร

4. น้ำที่ใช้รดควรมีค่า pH เท่ากับ 7 (pH balance)

5. อากาศภายในโรงเรือนสามารถถ่ายเทได้ดี

การดูแลรักษา

โรคของเห็ดนางฟ้าภูฏาน ได้แก่ ราดำ ราเขียว ราฝุ่นและแมลงหวี่เห็ดปีกดำ ส่วนมากราดำและราเขียวจะเกิดในช่วงของการพักเชื้อ ทั้งนี้อาจจะติดเชื้อมาจากช่วงของการใส่เชื้อเห็ด เนื่องจากสถานที่ใส่เชื้อมีลมแรงหรือสถานที่ไม่สะอาด หรืออาจเกิดจากขั้นตอนการนึ่งก้อนเชื้อที่ใช้ความร้อนไม่สูงพอ

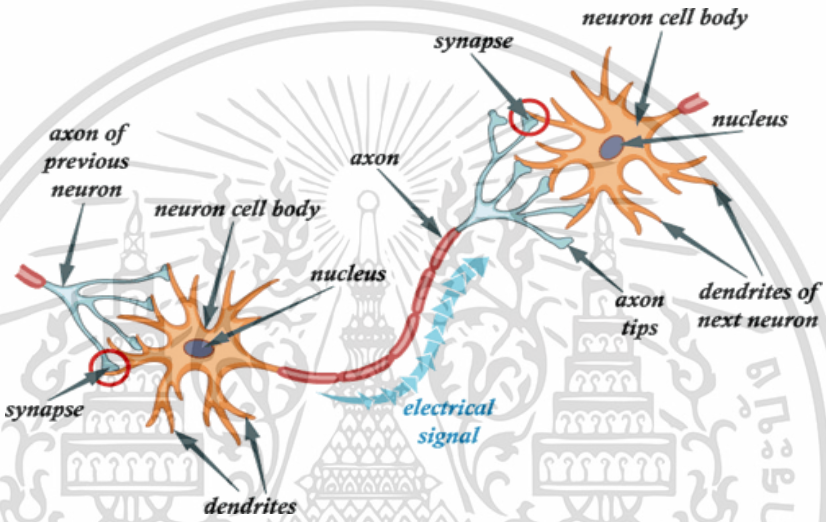
การแก้ไขคือ ควรนึ่งก้อนเพาะเชื้อเห็ดที่อุณหภูมิประมาณ 75-100 องศาเซลเซียส เป็นเวลา 3-4 ชั่วโมง และสถานที่ที่ใช้ทำการใส่เชื้อที่สะอาด ไม่มีลมพัด จะสามารถช่วยป้องกันการเกิดโรคได้ (บุญส่ง วงศ์เกรียงไกร, 2000)

2.2 โครงข่ายประสาทเทียม (Artificial Neural Networks)

โครงข่ายประสาทเทียมหรือ Artificial Neural Networks (ANN) คือ แบบจำลองทางคณิตศาสตร์หรือแบบจำลองทางคอมพิวเตอร์ซึ่งมีรากฐานมาจากเทคโนโลยีปัญญาประดิษฐ์ (Artificial Intelligence, AI) เป็นชุดของขั้นตอนวิธีที่พยายามจดจำชุดข้อมูลผ่านกระบวนการทำงาน ที่เลียนแบบมาจากการทำงานของระบบสมองมนุษย์สำหรับประมวลผลข้อมูลสารสนเทศ สามารถเรียนรู้ คิดวิเคราะห์ข้อมูลที่ซับซ้อนและปรับตัวให้รับรู้สภาพที่เปลี่ยนแปลง เพื่อใช้ในการช่วยตัดสินใจหรือสร้างผลลัพธ์ขั้นสุดท้าย

2.2.1 เซลล์ประสาท (Neurons)

เซลล์ประสาทในทางชีววิทยา (Neurons or Nerve cells) เป็นหน่วยพื้นฐานของระบบประสาทและสมองของมนุษย์ (ดร.วราวุธ วุฒิวณิชย, 2002) ทำหน้าที่ในการจดจำ คิด นำความรู้และประสบการณ์ต่างๆ ที่บันทึกไว้ในสมองมาใช้ตัดสินใจในชีวิตประจำวัน โดยแต่ละเซลล์ประสาทจะเชื่อมต่อกับเซลล์ประสาทอื่นๆ มากถึง 200,000 เซลล์ ตามหลักประสาทวิทยา (Neuroscience) จะแบ่งการทำงานออกเป็น 4 ส่วน ดังรูปที่ 2.2 ซึ่งได้แก่



รูปที่ 2.2 เซลล์ประสาทของมนุษย์

ที่มา https://simplebiologyy.blogspot.com/2014/08/conduction-of-nerve-impulse.html

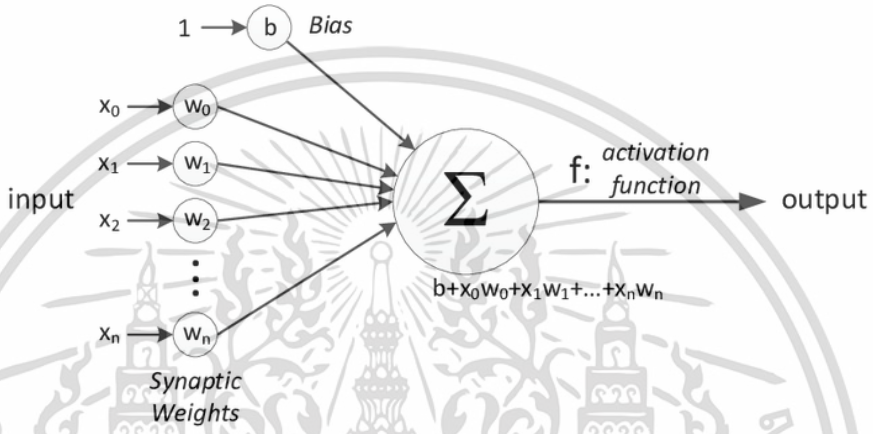
1. Dendrite คือ ส่วนที่ทำหน้าที่ในการรับข้อมูล (Accept Input) กระแสประสาทจากเซลล์อื่นในรูปของสัญญาณเข้ามา

2. Cell Body (Soma) คือ ส่วนที่ทำหน้าที่รับข้อมูลมาจาก Dendrite มาทำการประมวลผลเบื้องต้น (Process Input) เป็นคำสั่งในรูป Action Potential

3. Axon คือ ส่วนที่แปลงข้อมูลที่ประมวลผลแล้วจาก Cell Body เป็นผลลัพธ์ที่ต้องการ

4. Synapse คือ เส้นประสาทที่ทำหน้าที่เชื่อมต่อเพื่อสื่อสารกับเส้นประสาทอื่นๆ ในระบบสมองเพื่อช่วยกันสร้างผลลัพธ์ขั้นสุดท้าย

ซึ่งจากแนวคิดการทำงานของเซลล์ประสาทนี้ นำไปสู่การพัฒนาสร้างเป็นข่ายประสาทเทียม (Single Layer Neural Network) หรือที่เรียกว่า เพอร์เซปตรอน (Perceptron) ใน ค.ศ. 1957 โดย Frank Rosenblatt ที่ Cornell Aeronautical Laboratory มีการทำงานแบบชั้นเดียวคืออินพุตจะส่งตรงไปยังเอาต์พุตโดยผ่านชุดค่าน้ำหนัก นิวรอนแต่ละตัวจะคำนวณผลรวมของผลคูณระหว่างค่าอินพุตกับค่าน้ำหนักของแต่ละตัว โดยใช้ฟังก์ชันกระตุ้น (Activation Function) เพื่อสร้างผลลัพธ์เป็นเอาต์พุต ดังรูปที่ 2.3

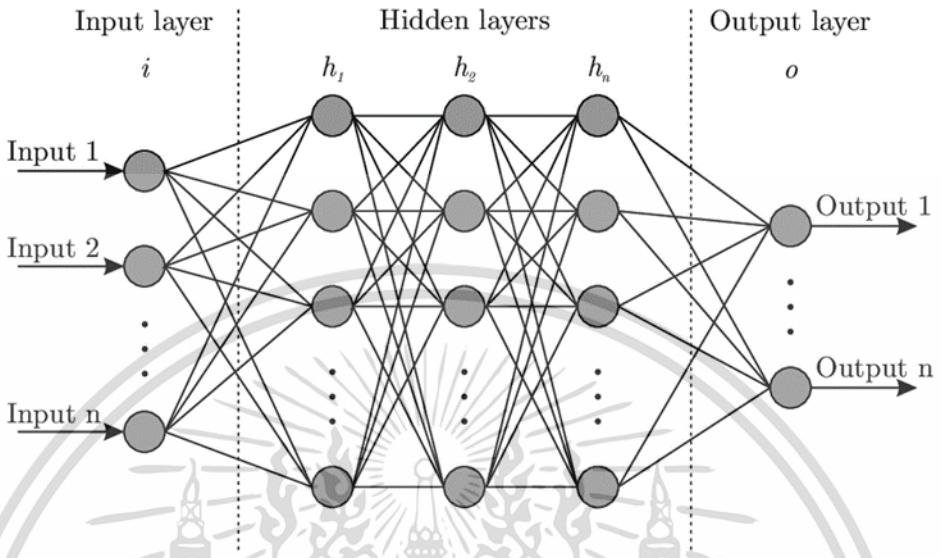


รูปที่ 2.3 เพอร์เซปตรอน

ที่มา https://content.iospress.com/articles/journal-of-high-speed-networks/jhs594

แต่อย่างไรก็ตาม Frank Rosenblatt ยังไม่สามารถสอนให้เพอร์เซปตรอนที่จำลองการทำงานอย่างง่ายๆ นี้ ให้สามารถเรียนรู้ จดจำรูปแบบต่างๆ หลายแบบได้ จนกระทั่งในปี ค.ศ.1980 มีการพัฒนาจนเกิดเป็นข่ายงานประสาทหลายชั้น (Multi-layer Neural Network)

2.2.2 การออกแบบ Artificial Neural Network



รูปที่ 2.4 โครงสร้างของ Artificial Neural Network

ที่มา https://www.researchgate.net/figure/Artificial-neural-network-architecture-ANN-i-h-1-h-2-h-n-o_fig1_321259051

1. กำหนดจำนวนชั้นของนิวรอน โดยชั้นของนิวรอนสามารถแบ่งออกได้ดังนี้ (ดร.วราวุธ วุฒิวณิชย, 2002)

- Input Layer นิวรอนในชั้นนี้ทำหน้าที่ในการรับข้อมูลเข้ามา และส่งต่อให้ Hidden Layer โดยชั้นอินพุตนี้จะมีเพียงชั้นเดียวเท่านั้น

- Hidden Layer มีหน้าที่รับข้อมูลจากชั้นก่อนหน้า มีสมการที่ใช้ในการคำนวณเพื่อประมวลผลข้อมูลจากอินพุตส่งต่อไปยังชั้นเอาต์พุต

- Output Layer นิวรอนในชั้นนี้จะรอรับค่าจาก Hidden Layer ชั้นสุดท้าย แล้วบอกว่าผลลัพธ์ของการประมวลผลคืออะไร

2. กำหนดการเชื่อมโยงระหว่างนิวรอนแต่ละชั้นโดยให้ทุกนิวรอนในชั้นที่ 1 เชื่อมโยงกับทุกนิวรอนในชั้นที่ 2 และนิวรอนในชั้นที่ 2 เชื่อมโยงกับนิวรอนทุกตัวในชั้นที่ 3 เชื่อมโยงแบบนี้ทีละชั้นจนถึง Output Layer

3. สอนให้โครงข่ายประสาทเทียมเรียนรู้ความสัมพันธ์ระหว่างข้อมูล และผลลัพธ์ที่ต้องการเพื่อสร้างค่าน้ำหนัก (Weight) และค่าความเบี่ยงเบน (Bias) ที่เหมาะสม โดยใช้ชุดข้อมูลฝึกอบรม (Training Data Set)

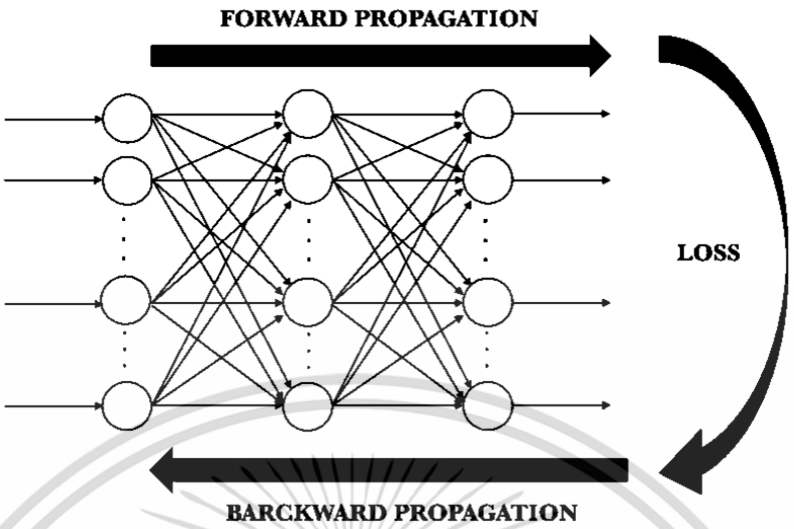
2.2.3 การเรียนรู้ของ Machine Learning

ความสามารถในการเรียนรู้ของ ANN (Nattapon Rakthong, 2018) ขึ้นอยู่กับการออกแบบโครงสร้างและวิธีที่ใช้ในการเรียนรู้ การเชื่อมโยงของแต่ละนิวรอนจะมีค่าน้ำหนัก (Weight) แต่ละนิวรอนเรียนรู้โดยการปรับค่าน้ำหนัก เพื่อให้ได้การทำนายผลลัพธ์เอาต์พุตที่มีความคลาดเคลื่อนน้อยที่สุด โดยสามารถแบ่งการเรียนรู้ออกเป็น 2 ประเภทหลัก คือ

1. การเรียนรู้แบบไม่มีผู้สอน (Unsupervised Learning) เป็นกลุ่ม Algorithm ที่ไม่มีการสอนอย่างชัดเจนว่าในการประมวลผลแล้วได้ผลลัพธ์ออกมา ผลลัพธ์นั้นถูกหรือผิด มักใช้ในการแบ่งกลุ่มโดยแบ่งแบบไม่มีกฎตายตัว ว่าต้องแบ่งกลุ่มตามเกณฑ์อะไร จะทำการแบ่งเป็นกลุ่มตามจำนวนกลุ่มที่ต้องการ ประเภทของการเรียนรู้แบบไม่มีผู้สอนมี 2 ประเภทหลักคือ การแบ่งกลุ่มข้อมูล (Clustering) และการลดมิติของข้อมูล (Dimension Reduction)

2. การเรียนรู้แบบมีผู้สอน (Supervised Learning) เป็นกลุ่ม Algorithm ที่จะสอนคอมพิวเตอร์ด้วยการฝึกอบรม (Training) โดยจะมีชุดข้อมูลไว้สำหรับฝึกอบรมให้คอมพิวเตอร์จดจำ เรียนรู้เพื่อใช้ในการประมวลผลได้อย่างถูกต้องแม่นยำเมื่อใส่ข้อมูลชุดใหม่ (Testing) เข้าไป การเรียนรู้แบบมีผู้สอนมี 2 ประเภทได้แก่ การแบ่งแยกประเภท (Classification) และการถดถอย (Regression) ซึ่งในการทดลองนี้ทั้งวิธี ANN, KNN, NCC และ SVM จัดอยู่ในกลุ่ม Supervised Learning ทั้งหมด

สำหรับวิธีที่ ANN จะเป็นการทำงานแบบ Feed Forward คือ ประมวลผลแบบไปข้างหน้าจากอินพุตไปยังเอาต์พุต จนกระทั่งในปี ค.ศ. 1986 Rumelhart ได้ตีพิมพ์บทความทางวิชาการเกี่ยวกับ Algorithm ที่ใช้วิธี Backpropagation ซึ่งคือการปรับค่าน้ำหนักในแต่ละจุดให้ดีขึ้นเพื่อให้ได้ค่าผลลัพธ์ที่แม่นยำมากขึ้นกว่าเดิม โดยการใช้การทำนายโดยการคำนวณค่าเอาต์พุต และใช้ค่าความผิดพลาดของการทำนายในการปรับค่าน้ำหนักด้วยวิธี Gradient Descent โดยเริ่มจากชั้นเอาต์พุตและชั้นถัดมาทางด้านซ้าย ดังรูปที่ 2.5



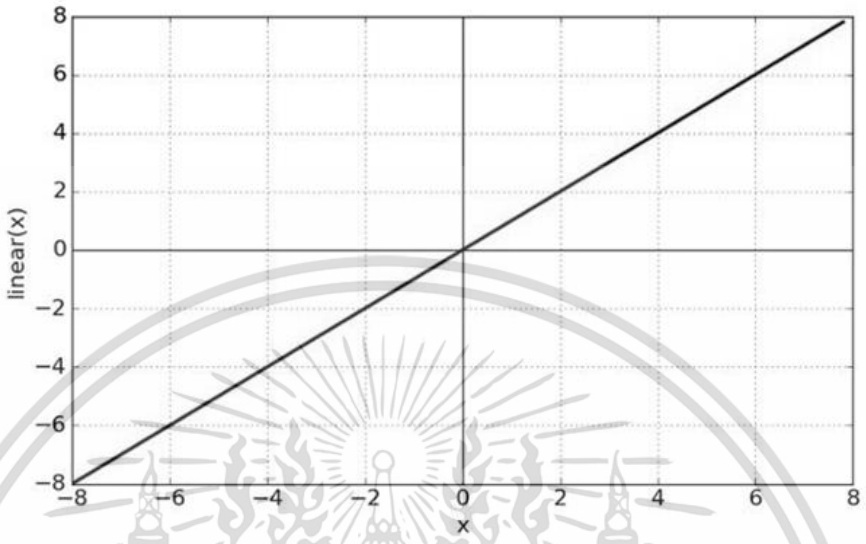
รูปที่ 2.5 แนวคิดการทำงานของ Backpropagation

ที่มา https://towardsdatascience.com/introduction-to-artificial-neural-networks-ann-1aea15775ef9

2.2.4 ฟังก์ชันการกระตุ้น (Activation Function)

ฟังก์ชันการกระตุ้นคือ ฟังก์ชันที่ใช้ในการคำนวณเพื่อหาเอาต์พุตของแต่ละนิวรอน โดยในการทดสอบนั้นจะสามารถเลือกชนิดของฟังก์ชันของแต่ละชั้นได้ โดยฟังก์ชันที่ผู้จัดทำได้เลือกมาทดสอบได้แก่

1. Linear Activation Function

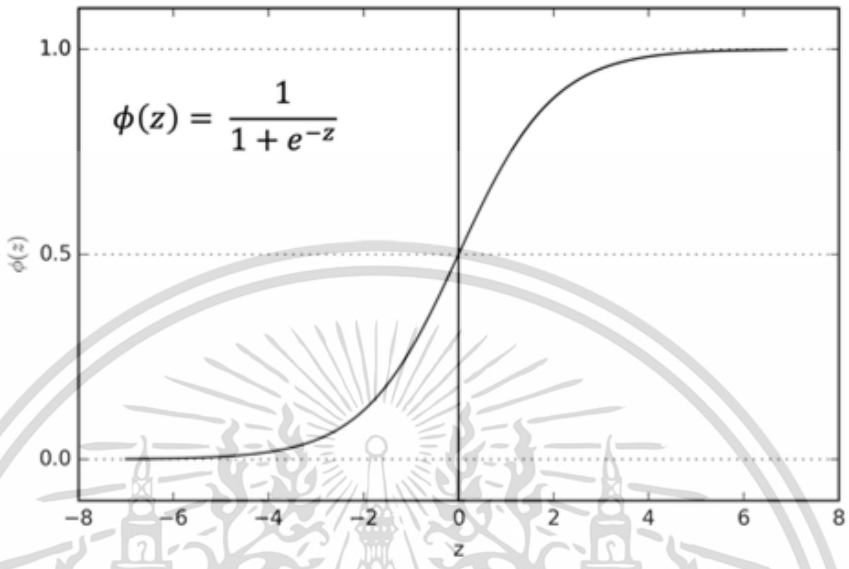


รูปที่ 2.6 Linear Activation Function

ที่มา https://zamronijr.github.io/blog/tech/2018/04/23/intro-to-ann.html

เป็นฟังก์ชันเชิงเส้นที่ง่ายต่อการใช้แก้ปัญหา แต่ไม่เป็นที่นิยมนักเนื่องจากความซับซ้อนของข้อมูลที่ยิ่งมากเท่าไหร่ ก็ยิ่งส่งผลกระทบต่อประสิทธิภาพในการทำงานของฟังก์ชัน

2. Sigmoid Activation Function



รูปที่ 2.7 Sigmoid Activation Function

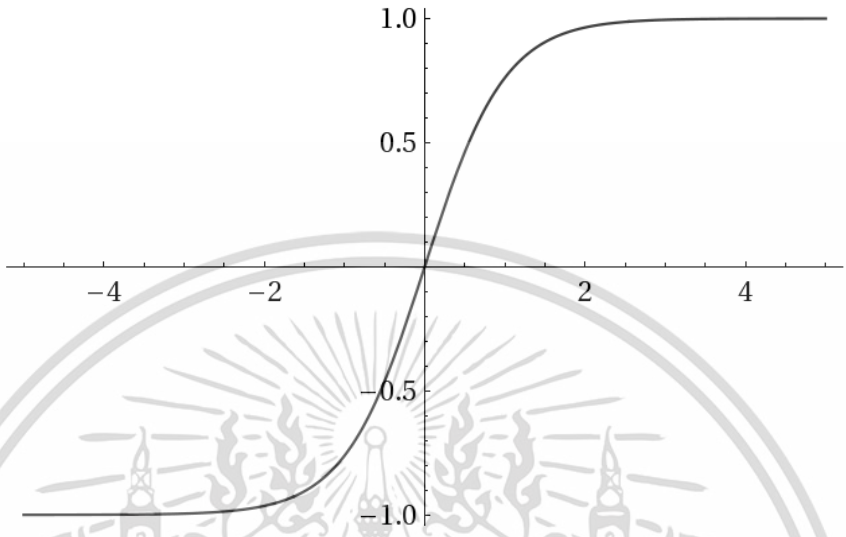
ที่มา https://medium.com/mmp-li/deep-learning-แบบสามัญชน-ep-2-optimization-activation-function-เรียนกันสบายๆสไตส์ชิลๆ-9feb5a87e3b2

เรียกอีกชื่อหนึ่งว่า Logistic Function เป็นฟังก์ชันที่เป็นเส้นโค้งรูปตัว S เอาต์พุตมีค่าอยู่ระหว่าง 0 ถึง 1 เหมาะกับใช้ในงานที่ต้องการผลลัพธ์เป็นความน่าจะเป็น (Probability) โดยกำหนดให้กลุ่มข้อมูลที่ต้องการเท่ากับ 1 และกลุ่มข้อมูลที่ไม่ต้องการเท่ากับ 0 (Sagar Sharma, 2017)

ข้อเสียของ Sigmoid Activation Function คือ Neural Network อาจจะติดขัดในขณะที่สร้างการเรียนรู้ของแบบจำลอง เนื่องจากแบบจำลองจะลู่เข้า (Converge) ช้า ทำให้เรียนรู้ได้ช้า และหากอินพุตอยู่นอกระยะ -5 ถึง 5 ความชันจะเข้าใกล้ 0 ซึ่งทำให้ Gradient เล็กลงเรื่อยๆ และลู่เข้าหาค่าศูนย์ (ปลายทั้งสองข้างแบนราบ) เกิดปัญหา Vanishing Gradient คือ ค่าน้ำหนักไม่ถูกอัพเดทอีกต่อไป

นอกจากนี้ เนื่องจากฟังก์ชันมีค่าอยู่ระหว่าง 0 ถึง 1 ซึ่งค่าเฉลี่ยกึ่งกลางไม่เท่ากับ 0 ทำให้ค่า Gradient สลับไปมาระหว่างการเรียนรู้ของแบบจำลอง

3. Hyperbolic Tangent (Tanh) Activation Function



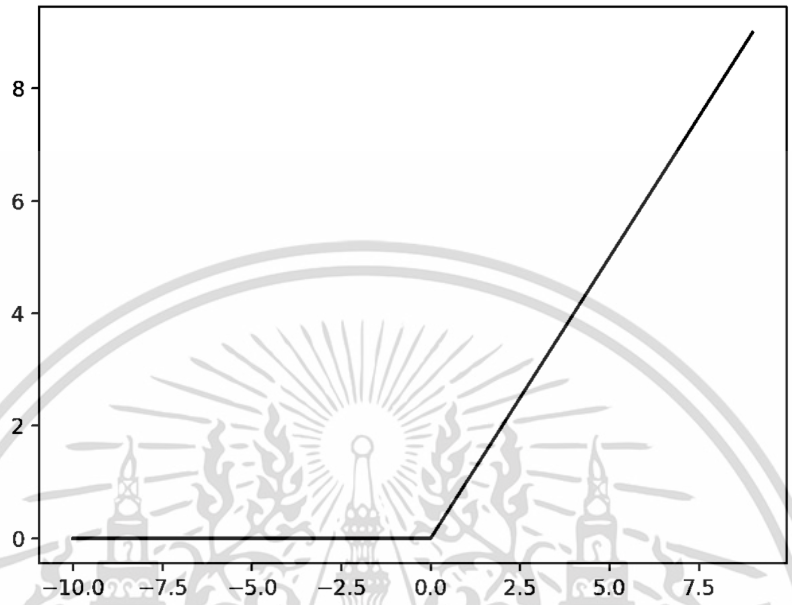
รูปที่ 2.8 Hypoberlic Tangent (Tanh) Activation Function

ที่มา https://www.researchgate.net/figure/The-Hyperbolic-Tangent-Activation-Function_fig1_224535485

มีค่าของฟังก์ชันคล้ายกับ Sigmoid แต่เอาต์พุตจะมีค่าตั้งแต่ -1 ถึง 1 ทำให้มีข้อดีกว่า Sigmoid เพราะมีค่ากึ่งกลางอยู่ระหว่าง 0 ทำให้ Gradient ไม่สลับไปมา และมีความชันมากกว่า Sigmoid ทำให้ Gradient มากกว่าและเรียนรู้ได้เร็วกว่า แต่ยังคงถือว่า Converge ช้าและไม่สามารถใช้ในงานที่ต้องการผลลัพธ์เป็นความน่าจะเป็นได้ นิยมใช้ในงาน Classification ที่มีเพียงแค่ 2 กลุ่ม (Sagar Sharma, 2017)

และหากอินพุตอยู่นอกระยะ -3 ถึง 3 ความชันจะเข้าใกล้ 0 และทำให้เกิดปัญหา Vanishing Gradient เช่นเดียวกับ Sigmoid

4. Rectified Linear Unit (ReLU) Activation Function



รูปที่ 2.9 Rectified Linear Unit (ReLU) Activation Function

ที่มา https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/

เป็นหนึ่งในฟังก์ชันที่นิยมนำมาใช้ในการสร้างแบบจำลองการเรียนรู้แบบ ANN เนื่องจากเป็นฟังก์ชันเส้นตรงที่ถูกปรับแก้เอาต์พุตมีค่าเพียงครึ่งเดียวคือ ตั้งแต่ 0 ถึงอนันต์ จัดว่าเป็นฟังก์ชันที่เรียบง่าย แต่มีประสิทธิภาพกว่า Activation Function อื่น เนื่องจากถ้าอินพุตเป็นบวก ความชันจะเท่ากับ 1 ตลอดกาล ส่งผลให้ไม่เกิด Vanishing Gradient ทำให้การเรียนรู้ของแบบจำลองเร็วขึ้นมาก (Sagar Sharma, 2017)

ซึ่งหากไม่มีฟังก์ชันการกระตุ้นแล้ว แบบจำลองโครงข่ายประสาทเทียมจะไม่สามารถเรียนรู้และสร้างแบบจำลองของข้อมูลที่มีความซับซ้อน เช่น รูปภาพ วิดิโอ เสียงหรือข้อความยาวๆ เช่น คำบรรยาย

2.3 K–Nearest Neighbors (KNN)

KNN เป็นวิธีหนึ่งที่ใช้ในการจัดกลุ่มข้อมูลที่อยู่ในกลุ่ม Supervised Learning โดยใช้วิธีการค้นหาเพื่อนบ้านที่ใกล้ที่สุด เพื่อตัดสินใจว่าข้อมูลที่ใส่เข้าไปจะตรงกับเงื่อนไขหรือกรณีใด โดยจะมีกลุ่มข้อมูลหนึ่งอยู่ก่อนแล้ว และจะเลือกค่า K หรือจำนวนเพื่อนบ้านที่ใกล้ที่สุดที่มีลักษณะเหมือน หรือใกล้เคียงกับข้อมูลใหม่ที่เข้ามามากที่สุด โดยจะวัดจากระยะห่างของแต่ละกรณีจากแต่ละกลุ่มที่มีลักษณะใกล้เคียงกับข้อมูลใหม่ที่เข้ามา

2.3.1 หลักการทำงานของขั้นตอนวิธี K-Nearest Neighbors

KNN จะดูจากจำนวนข้อมูลบริเวณรอบๆ ที่ใกล้ที่สุด (Nearest Data) ของข้อมูลที่ต้องการพิจารณา ว่าในจำนวน K เพื่อนบ้านที่ใกล้ที่สุดนั้น มีจำนวนข้อมูลจากกลุ่มไหนอยู่ใกล้ที่สุดเยอะกว่ากัน จากนั้น KNN จะอนุมานให้ข้อมูลใหม่ที่ใส่เข้าไปเป็นข้อมูลประเภทเดียวกับกลุ่มข้อมูลนั้นๆ โดย KNN มีขั้นตอนดังนี้

1. กำหนดขนาดของ K
2. คำนวณระยะห่างของข้อมูลระหว่างข้อมูลที่ต้องการพิจารณากับกลุ่มข้อมูลตัวอย่าง
3. จัดเรียงลำดับของระยะห่างและเลือกพิจารณาข้อมูลที่อยู่ใกล้ที่สุดตามจำนวน K ที่กำหนดไว้
4. พิจารณาจำนวนชุดข้อมูลที่ใกล้ที่สุดจำนวน K ตัว แล้วพิจารณาว่ากลุ่มข้อมูลไหนอยู่ใกล้ข้อมูลที่พิจารณามากที่สุด
5. ตัดสินใจเลือกกลุ่มให้กับข้อมูลที่พิจารณา

2.3.2 ตัวอย่างการทำงานของขั้นตอนวิธี K-Nearest Neighbors

1. สมมติให้มีกลุ่มข้อมูล 2 กลุ่ม และกำหนดให้ค่า K=3

รูปที่ 2.10 กลุ่มข้อมูลที่ใช้เป็น Training Data

2. จากนั้นใส่ข้อมูลใหม่เข้ามา (สีม่วง) ซึ่งเป็นข้อมูลที่ต้องการพิจารณาว่าควรอยู่ในกลุ่มใด

รูปที่ 2.11 มีข้อมูลใหม่ที่ต้องการพิจารณาเข้ามา

3. คำนวณระยะห่างของข้อมูลที่อยู่ใกล้ข้อมูลใหม่มากที่สุด (Nearest Neighbors) 3 ข้อมูลตามค่า K ที่กำหนด พบว่าข้อมูลที่ใกล้ที่สุด 3 ข้อมูลได้แก่ สีเหลือง 1 ข้อมูล และสีแดง 2 ข้อมูล

รูปที่ 2.12 หาเพื่อนบ้านที่ใกล้ที่สุด 3 ตัว

4. จากนั้นจึงจัดให้ข้อมูลใหม่ที่พิจารณาอยู่ในกลุ่มข้อมูลสีแดง

รูปที่ 2.13 จัดกลุ่มให้ข้อมูล

2.3.3 ข้อควรระวังในการใช้วิธี KNN

1. ค่า K ควรมีค่าเป็นเลขคี่ เนื่องจากหากมีจำนวนกลุ่มเป็นเลขคู่ อาจเกิดสถานการณ์ที่ทำให้ไม่สามารถจัดแบ่งกลุ่มได้ เพราะจำนวนเพื่อนบ้านที่ใกล้ที่สุดจากทั้งสองกลุ่มมีค่าเท่ากัน ดังรูปที่ 2.14

รูปที่ 2.14 ปัญหาเมื่อ *K* เป็นเลขคู่สำหรับกลุ่มที่มีจำนวนคู่

2. ค่า K ที่กำหนดจะต้องไม่มีค่าเท่ากับค่าผลคูณของจำนวนกลุ่ม เช่น จากรูปที่ 2.15 เมื่อมีชุดข้อมูลทั้งหมด 3 กลุ่ม หากกำหนดให้ค่า K เท่ากับ 6 ซึ่งเป็นผลคูณของ 3 อาจเกิดกรณีที่มีจำนวนเพื่อนบ้านที่ใกล้ที่สุดจำนวนเท่ากันจากแต่ละกลุ่ม ทำให้ไม่สามารถจัดแบ่งกลุ่มได้ ดังรูปที่ 2.15

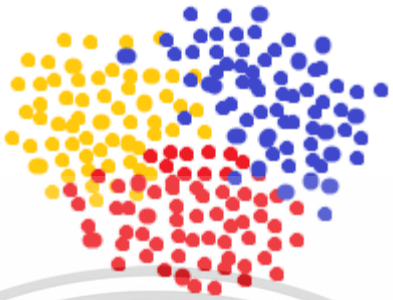
รูปที่ 2.15 ปัญหาเมื่อมีข้อมูล 3 กลุ่ม และ *K=6* ซึ่งเป็นผลคูณของ 3

2.4 Nearest Centroid Classifier (NCC)

เป็นวิธีหนึ่งในการจัดแบ่งกลุ่มข้อมูลที่เรียบง่าย แต่มีประสิทธิภาพ สามารถเรียกอีกอย่างได้ว่าเป็นการแบ่งกลุ่มแบบ Mean Difference โดยหลักการทำงานคือ พิจารณากลุ่มข้อมูลตัวอย่าง หาจุดกึ่งกลาง (ค่าเฉลี่ย) ของกลุ่มข้อมูลตัวอย่างแต่ละกลุ่ม เมื่อมีข้อมูลใหม่เข้ามา พิจารณาหาระยะทางว่าชุดข้อมูลใหม่ที่เข้ามาอยู่ใกล้กับจุดกึ่งกลางของกลุ่มข้อมูลใดมากที่สุด ซึ่งจะอนุมานให้ข้อมูลที่ต้องการพิจารณาเป็นข้อมูลในกลุ่มนั้นๆ

2.4.1 ตัวอย่างการทำงานของขั้นตอนวิธี Nearest Centroid

1. สมมติให้มีข้อมูลอยู่ 3 กลุ่มคือ เหลือง แดง และน้ำเงิน

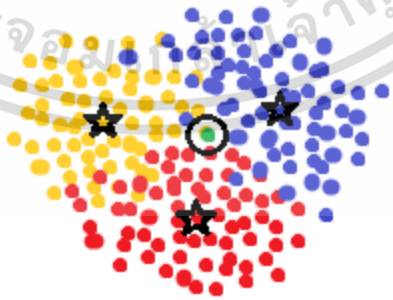


รูปที่ 2.16 แสดงกลุ่มข้อมูล 3 กลุ่ม

2. ใส่ข้อมูลใหม่ที่ต้องการพิจารณาเข้ามา ซึ่งคือสีเขียว

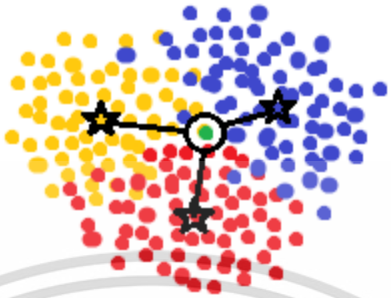
รูปที่ 2.17 มีข้อมูลใหม่ที่ต้องการพิจารณาเข้ามา

3. หาจุดกึ่งกลางหรือค่าเฉลี่ยของกลุ่มข้อมูลทั้ง 3 กลุ่มว่าอยู่ตำแหน่งใด



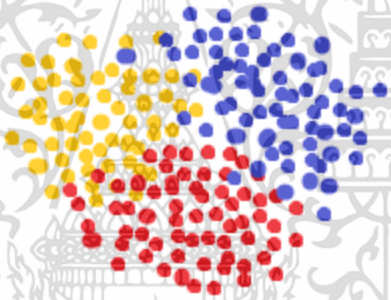
รูปที่ 2.18 หา Centroid ของแต่ละกลุ่มข้อมูล

4. หาระยะทางระหว่างข้อมูลใหม่ที่ต้องการพิจารณากับจุดกึ่งกลางของกลุ่มข้อมูล ดูว่าอยู่ใกล้กับจุดกึ่งกลางของกลุ่มใดที่สุด ซึ่งจะเห็นว่าข้อมูลใหม่อยู่ใกล้กับจุดกึ่งกลางของกลุ่มข้อมูลสีน้ำเงินมากที่สุด



รูปที่ 2.19 หา Nearest Centroid

5. จัดให้ข้อมูลใหม่อยู่ในกลุ่มข้อมูลสีน้ำเงิน



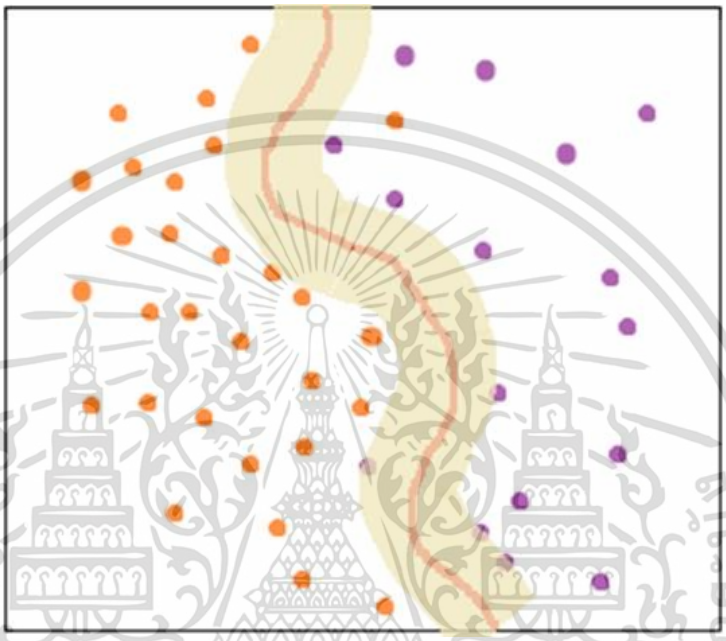
รูปที่ 2.20 จัดกลุ่มข้อมูล

2.5 Support Vector Machine (SVM)

เป็นแบบจำลองทางคณิตศาสตร์ที่จัดอยู่ในกลุ่ม Supervised Learning เป็น Algorithm ที่นิยมใช้กันอย่างกว้างขวางในงานประมวลผลภาพ (Image Processing) และงานที่เกี่ยวข้องกับการจดจำรูปแบบหรืองานจำแนกประเภท (Classification) โดยใช้หลักการจำแนกข้อมูลออกเป็นสองกลุ่ม (Binary Classification) ซึ่งมีประสิทธิภาพในการจำแนกข้อมูลที่มีมิติจำนวนมากได้ โดยอาศัยการหาเส้นแบ่งการตัดสินใจ (Decision Boundary)

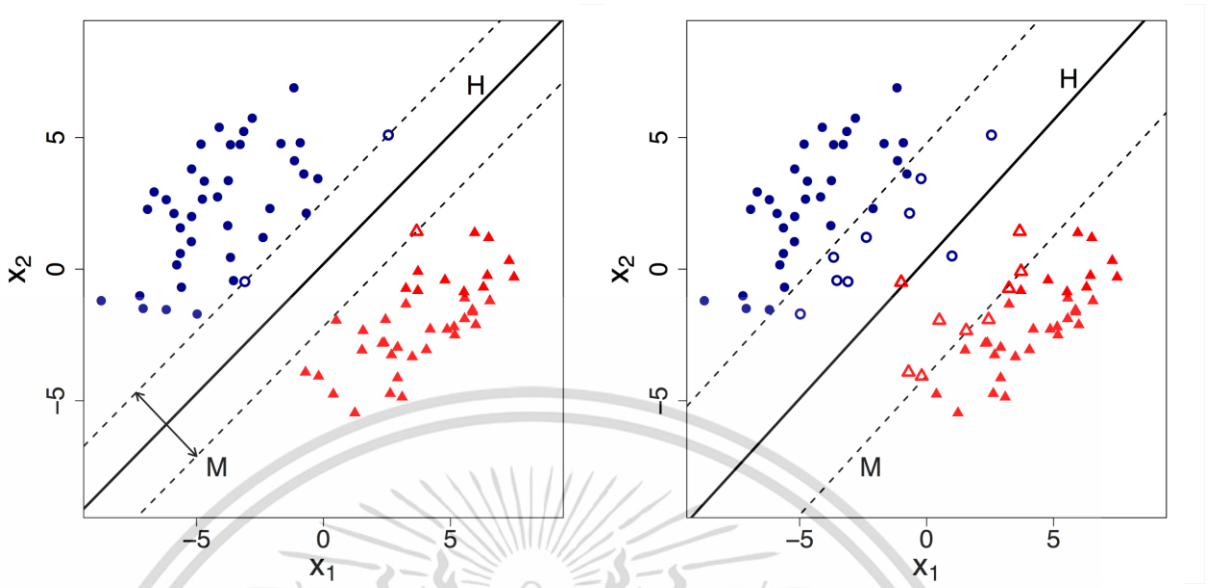
2.5.1 หลักการทำงานของของขั้นตอนวิธี Support Vector Machine

หลักการของ SVM คือ การให้อินพุตที่ใช้ฝึกอบรมเป็นเวกเตอร์ในพื้นที่ N มิติ จากนั้นจะทำการสร้างเส้นแบ่งหรือ Hyperplane เช่น ถ้าใน 2 มิติ จะเป็นเส้นแบ่งที่อยู่ในระนาบ xy และถ้า 3 มิติ จะเป็นระนาบที่อยู่ในมิติ xyz ซึ่งใช้ในการแยกข้อมูลออกเป็นกลุ่มต่างๆ ดังรูปที่ 2.21



รูปที่ 2.21 ตัวอย่างการแบ่งข้อมูลด้วยวิธี SVM แบบ 2 มิติ

จากรูปที่ 2.21 มีข้อมูลบางส่วนปะปนกันอยู่บ้าง และมีบางข้อมูลอยู่บริเวณขอบของเส้นแบ่ง โดยจุดข้อมูลที่อยู่ขอบของเส้นแบ่งจะเรียกว่า เวกเตอร์ค้ำยัน (Support Vector) และแม้จะมีข้อมูลที่ปะปนกันอยู่บ้างแต่สามารถแบ่งข้อมูลออกเป็นสองกลุ่มได้อย่างสมเหตุสมผล กรณีที่มีข้อมูลบางส่วนเข้ามาอยู่ในพื้นที่เส้นแบ่ง ซึ่งเป็นข้อมูลจำนวนน้อยจึงเลือกที่จะไม่พิจารณาข้อมูลนั้นจะเรียกว่า ขอบนุ่ม (Soft Margin) ส่วนแบบที่ไม่ยอมให้มีข้อมูลใดๆ อยู่บนขอบและแบ่งแบบไม่มีการปะปนของข้อมูลเลย เรียกว่า ขอบแข็ง (Hard Margin) ดังรูปที่ 2.22



รูปที่ 2.22 SVM แบบ Hard Margin และ Soft Margin

ที่มา https://www.surveypractice.org/article/2715-using-support-vector-machines-for-survey-research

บทที่ 3

วิธีการทดลอง

3.1 วิธีการดำเนินงาน

1. ศึกษาค้นคว้าทฤษฎีและเอกสารที่เกี่ยวข้อง
2. ศึกษาข้อมูลจากเกษตรกรที่ฟาร์มเห็ดนางฟ้าภูฏาน
3. ถ่ายรูปเก็บข้อมูลตัวอย่างก้อนเพาะเชื้อเห็ดนางฟ้าภูฏาน
4. คัดเลือกวิธีที่ใช้ในการคัดแยกคุณภาพก้อนเพาะเชื้อเห็ดนางฟ้าภูฏาน
5. ศึกษาการทำงานของโปรแกรมและเริ่มต้นเขียนโปรแกรม
6. ทดลองใช้โปรแกรมคัดแยกคุณภาพก้อนเพาะเชื้อเห็ดนางฟ้าภูฏาน
7. ตรวจสอบประสิทธิภาพและปรับปรุงการทำงานของโปรแกรม

3.2 ซอฟต์แวร์ที่เกี่ยวข้อง

1. Python

เป็นภาษาเขียนโปรแกรมระดับสูงที่ใช้กันอย่างกว้างขวาง โดยถูกออกแบบมาให้เป็นภาษาที่อ่านง่าย ลดความซับซ้อนของโครงสร้างและไวยกรณ์ของภาษา การแปลงชุดคำสั่งที่เขียนให้เป็นภาษาเครื่อง Python มีการทำงานแบบ Interpreter คือเป็นการแปลชุดคำสั่งทีละบรรทัด เพื่อป้อนเข้าสู่หน่วยประมวลผลให้คอมพิวเตอร์ทำงานตามที่ผู้ใช้ต้องการ และโครงสร้างของภาษานั้นจะทำให้โปรแกรมเมอร์สามารถเข้าใจแนวคิดการเขียนโปรแกรม โดยใช้บรรทัดที่น้อยลงกว่าภาษาอื่นอย่าง C++ และ Java นอกจากนี้ภาษาโปรแกรม Python ยังสามารถนำไปใช้ในการเขียนโปรแกรมได้หลากหลายประเภท (General-purpose Language) ไม่ว่าจะเป็นการเขียนโปรแกรมที่มีโครงสร้างขนาดเล็กไปจนถึงโปรแกรมโครงสร้างขนาดใหญ่ เนื่องจากมีโครงสร้างในเขียนการภาษาที่เข้าใจง่าย

รูปที่ 3.1 โปรแกรม Python

การเริ่มต้นการพัฒนาของภาษา Python นั้นเริ่มในเดือนธันวาคม ค.ศ. 1989 โดย Guido Van Rossum โปรแกรมเมอร์ชาวดัตช์ ซึ่งทำงานอยู่ที่ Centrum Wiskunde & Informatica (CWI) สถาบันวิจัยทางด้านคณิตศาสตร์ และวิทยาการคอมพิวเตอร์ในเมืองอัมสเตอร์ดัม ประเทศเนเธอร์แลนด์ โดย Guido มีจุดประสงค์ในการพัฒนาโปรแกรมสำหรับผู้ดูแลระบบเพื่อใช้ในโครงการ Amoeba ซึ่งเป็นโครงการเกี่ยวกับระบบปฏิบัติการแบบกระจาย (Distributed Operating System) อย่างไรก็ตาม Guido ได้เล็งเห็นข้อจำกัดของภาษาโปรแกรม ABC, C และ Bourne Shell ทั้งเรื่องใช้เวลาในการพัฒนานาน และไม่ตรงกับเป้าหมายหลายประการ ดังนั้น Guido จึงได้ตัดสินใจเริ่มพัฒนาภาษาโปรแกรมระดับสูงขึ้นมาใหม่ เพื่อใช้งานเองเป็นงานอดิเรก โดยนำเอาข้อดีในภาษา ABC มาพัฒนาในภาษา Python รวมถึงได้พัฒนาส่วนอื่นๆ เพิ่มเติมเข้าไป และในเวลาต่อมาจึงได้เผยแพร่ Python 1.0 เวอร์ชันแรกในปี ค.ศ. 1994

3.3 ขั้นตอนการวางแผนและเก็บข้อมูล

การเก็บข้อมูลรูปภาพก้อนเพาะเชื้อเห็ดนางฟ้าภูฏาน

รูปที่ 3.2 การเก็บรูปภาพก้อนเพาะเชื้อเห็ดนางฟ้า

ตัวอย่างรูปก้อนเพาะเชื้อเห็ดแต่ละกลุ่มจากฟาร์มเห็ด รักเห็ดฟาร์ม จังหวัดสุพรรณบุรี โดยจะแบ่งคุณภาพออกเป็น 3 กลุ่มหลัก ได้แก่

กลุ่ม A : ก้อนเพาะเชื้อเห็ดที่ Mycelium เจริญเติบโตเต็มถุง

กลุ่ม B : ก้อนเพาะเชื้อเห็ดที่ Mycelium เจริญเติบโตไม่เต็มถุง แต่ยังสามารถนำไปเปิดดอกในโรงเรือนได้

กลุ่ม C : ก้อนเพาะเชื้อเห็ดที่พบปัญหาจากการติดเชื้อรา หรือมีการเจริญเติบโตของหนอนและแมลง ไม่สามารถนำไปเปิดดอกได้ แบ่งเป็น 4 กลุ่มย่อย ได้แก่ C1 ราเขียว, C2 ราดำ, C3 ราฝุ่น และ C4 มีหนอนหรือแมลง

รูปที่ 3.3 ก้อนเพาะเชื้อเห็ดกลุ่ม A และ B

รูปที่ 3.4 ก้อนเพาะเชื้อเห็ดกลุ่ม C1, C2, C3 และ C4 ตามลำดับ

3.4 การเตรียมข้อมูล

3.4.1 การดึงพื้นที่ที่สนใจ (Region of Interest) ออกจากพื้นหลัง

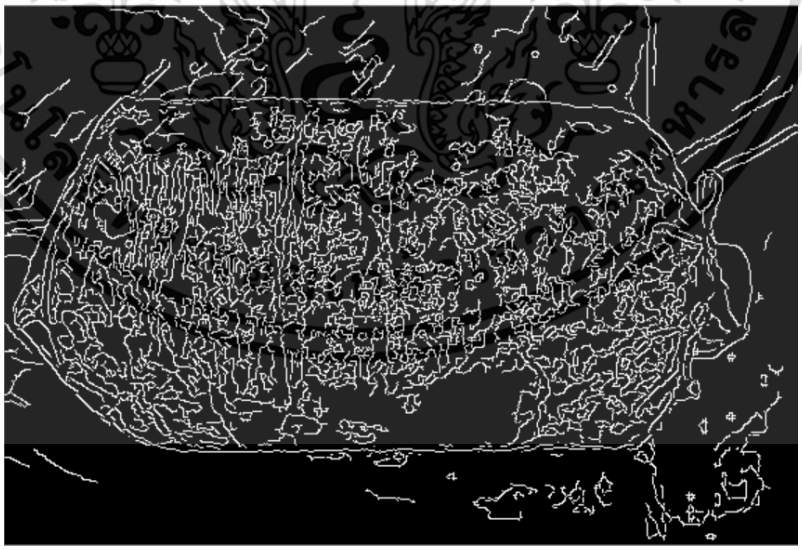
คือขั้นตอนการแยกรูปก้อนเพาะเชื้อเห็ดนางฟ้าภูฏานออกจากพื้นหลังด้วยเทคนิค Contour Detection และ Segmentation จากฟังก์ชันใน OpenCV ซึ่งเป็น Library Function สำหรับการเขียนโปรแกรม

โดยการดึงภาพก้อนเห็ดมีขั้นตอนดังนี้

1. เลือกรูปภาพที่ต้องการทำขึ้นมา

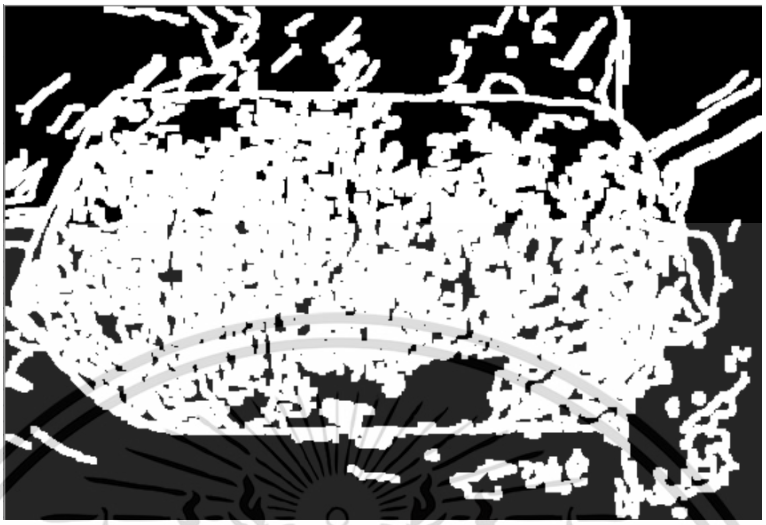
รูปที่ 3.5 ก้อนเพาะเชื้อเห็ดนางฟ้าภูฏาน

2. เปลี่ยนรูปภาพเป็น Grey Scale ด้วยฟังก์ชัน cv2.cvtColor() และเบลอรูปภาพเพื่อลด Noise ของภาพด้วยฟังก์ชัน GaussianBlur() จากนั้นจึงสร้างเส้นขอบรูปภาพด้วยฟังก์ชันตัวกรองความถี่สูง canny()



รูปที่ 3.6 การสร้างเส้นขอบรูปภาพ

3. เชื่อมต่อเส้นขอบที่ไม่ติดกันถึงกัน (Close Gap) ด้วยฟังก์ชัน dilate()



รูปที่ 3.7 การ Close Gap

4. ทำการลบเส้นขอบที่ไม่ควรจะมี เช่น เส้นขอบที่วงรอบวัตถุไม่กระชับหรือจุดสีเล็กๆ นอกวัตถุ ด้วยฟังก์ชัน erode()

รูปที่ 3.8 การลบเส้นขอบที่ไม่ต้องการ

5. เลือกแสดงเส้นที่วงรอบวัตถุเพียงเส้นเดียว เนื่องจากการสร้างเส้นขอบในขั้นตอนแรกนั้นจะเกิดเส้นขอบขึ้นมากมาย ทั้งเส้นขอบที่ซ้อนทับวัตถุหรือเส้นขอบที่วงรายละเอียดในวัตถุ จึงต้องนับจำนวนเส้นวงทั้งหมด และพื้นที่ที่เส้นวงแต่ละเส้นล้อมรอบด้วยฟังก์ชัน findContours()

6. ทำการบันทึกเส้นวงที่มีพื้นที่มากที่สุดด้วยฟังก์ชัน cv2.contourArea()

7. วาดเส้นวงนั้นล้อมรอบวัตถุด้วยฟังก์ชัน drawContours()

รูปที่ 3.9 วาดเส้นวงที่พื้นที่มากที่สุดซึ่งล้อมรอบวัตถุ

8. ระบายสีขาวในพื้นที่เส้นวงนั้น โดยใช้ฟังก์ชันเดียวกันกับข้อ 6 แต่ให้เลือกตัวแปรความหนาของเส้นวงเท่ากับ -1

รูปที่ 3.10 ระบายสีขาวในพื้นที่เส้นวง

9. นำรูปภาพที่ได้จากข้อ 8 รวมเข้ากับรูปภาพเดิมด้วยฟังก์ชัน bitwise_and() จะได้รูปภาพที่สามารถนำไปใช้สำหรับขั้นตอนต่อไปได้

รูปที่ 3.11 รูปภาพที่ทำการลบพื้นหลังเรียบร้อย

3.4.2 ขั้นตอนการประมวลผลภาพ (Pre-processing Step)

ขั้นตอนการประมวลผลภาพคือ ขั้นตอนการเตรียมข้อมูลให้พร้อมนำไปใช้ในการประมวลผล โดยการลดความซับซ้อนของข้อมูล การคัดกรองเอาเฉพาะคุณสมบัติเด่น การคัดแยกข้อมูลที่ไม่จำเป็นออกไป เพื่อเพิ่มความแม่นยำให้แก่แบบจำลอง ซึ่งในการทดลองนี้จะใช้ 2 เทคนิคในขั้นตอนการประมวลผลภาพ คือ

1. Principle Component Analysis (PCA) คือ การลดมิติของข้อมูลลงเพื่อลดความซับซ้อนและเวลาในการทำงานของโปรแกรม โดยใช้การเปลี่ยนแปลงเชิงเส้นในการหาแกนที่สำคัญของข้อมูล และกำจัดมิติของข้อมูลที่ไม่จำเป็นในการสร้างแบบจำลองออกไป โดยการทำ PCA อาศัยคำสั่ง PCA ใน Decomposition Module ของ scikit-learn (sklearn) ซึ่งเป็น Library สำหรับใช้ทำ Machine Learning และ Data Mining ใน Python ซึ่งในการทดลองเดิมมี 32,768 มิติ หลังจากทำ PCA แล้วเหลือ N มิติ โดยค่า N เป็นตัวแปรที่กำหนด

2. Scaling คือ การปรับลดค่าความแปรปรวน และค่าเฉลี่ยของข้อมูลให้อยู่ในมาตรฐานเดียวกัน คือ ค่าเฉลี่ยเท่ากับ 0 และค่าความแปรปรวนเท่ากับ 1 เพื่อให้ข้อมูลไม่กระจายตัวหรือชิดกันมากจนเกินไป ทำโดยใช้ฟังก์ชัน StandardScaler ใน Preprocessing Module ของ scikit-learn (sklearn)

3.5 เริ่มต้นศึกษาวิธีการเขียนซอฟต์แวร์

ได้ทำการศึกษาวิธีที่ใช้ในการจัดแบ่งกลุ่มข้อมูล และทำการทดลองเขียนโปรแกรม Python ออกมาโดยอิงหลักการจาก 4 วิธีที่สนใจ ได้แก่ Artificial Neural Network (ANN) , K-Nearest Neighbors (KNN), Nearest Centroid Classifier (NCC) และ Support Vector Machine (SVM) เพื่อหาวิธีที่จัดแบ่งข้อมูลได้แม่นยำที่สุด โดยเริ่มต้นจาก

1. ดาวน์โหลด Library ที่ต้องการใช้เข้ามา

2. ให้โปรแกรมอ่านค่ารูปก้อนเชื้อเห็ดแต่ละกลุ่ม

3. สร้างตัวแปรสำหรับฮีสโตแกรม

4. สร้างพื้นที่ว่างสำหรับใช้เก็บข้อมูลรูปภาพ

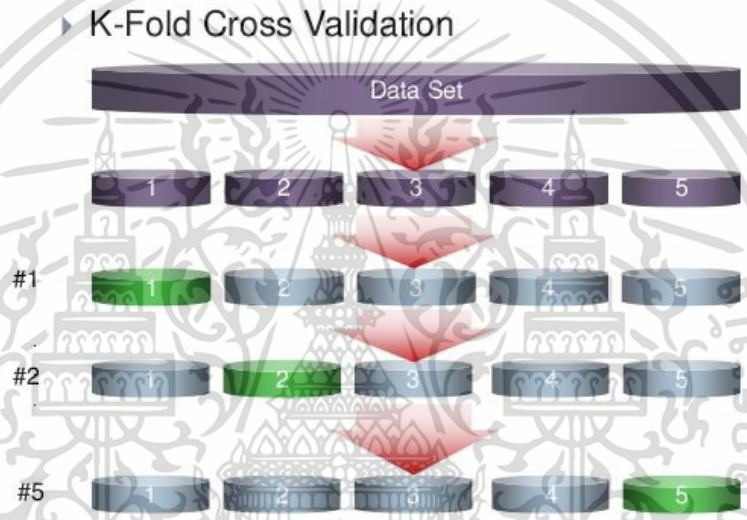
5. ทำการ Reshape ขนาดกลุ่มข้อมูล (Array)

6. สร้างแบบจำลองทางคณิตศาสตร์โดยใช้วิธีทาง Machine Learning

7. เขียนโปรแกรมสำหรับทดสอบประสิทธิภาพในการคัดแยกก้อนเพาะเชื้อเห็ด โดยใช้ 4 วิธีที่กล่าวถึงในหัวข้อก่อนหน้านี้ และทดสอบความแม่นยำด้วยวิธี K-Fold Cross Validation

3.6 วิธีการทำงานของโปรแกรม

เมื่อกำหนดให้ตัวแปรจำนวนกลุ่ม CV = 4 หมายความว่า สำหรับการทดสอบด้วย K-Fold Cross Validation จะมีการแบ่งตัวอย่างออกเป็น 4 กลุ่ม โดยใช้กลุ่มหนึ่งเป็นรูปสำหรับการ Test กับอีก 3 กลุ่มที่เหลือซึ่งเป็นกลุ่มสำหรับการ Train โดยจะทำการ Test และ Train วนสลับกันไปจนครบทั้ง 4 กลุ่ม และหาค่าความแม่นยำเฉลี่ย เช่น จากรูปที่ 3.12 กำหนดให้กลุ่มที่ 1 เป็นกลุ่มที่ใช้สำหรับการ Test และกลุ่มที่ 2, 3 และ 4 เป็นกลุ่มที่ใช้ในการ Train โดยจะนำกลุ่มที่ 1 ไปทดสอบกับอีก 3 กลุ่มที่เหลือทีละรูปจนครบ จากนั้นจึงเปลี่ยนให้กลุ่มที่ 2 เป็นกลุ่มสำหรับการ Test และนำไปทดสอบกับกลุ่ม 1, 3 และ 4 จากนั้นจึงทำเช่นเดียวกันกับกลุ่มที่ 3 และ 4



รูปที่ 3.12 การ Cross Validation

ที่มา https://www.slideshare.net/NontawatB/05-classification-1-decision-tree-and-rule-based-classification

3.7 การคัดแยกคุณภาพก้อนเพาะเชื้อเห็ดนางฟ้า

ในการสร้างแบบจำลอง จะต้องทำการติดตั้ง Package หรือ Library ที่จำเป็นสำหรับการใช้งานก่อน โดยในที่นี้ใช้ scikit-learn

1. เมื่อติดตั้ง Library เรียบร้อย ให้ทำการนำเข้า Library (Import Library) ที่จำเป็นเข้ามา

2. นำเข้าชุดข้อมูลเห็ดนางฟ้าภูฏานจากฐานข้อมูลโฟลเดอร์ที่มี (Import Data)

3. ตั้งค่าข้อมูล เป็นขั้นตอนที่กำหนดว่าให้คอลัมน์ใดคือ ตัวแปรอิสระ (Feature) และคอลัมน์ใดคือตัวแปรตาม (Class)

4. ทำการแบ่งข้อมูลออกเป็น 2 ส่วนคือ

1) Training Data เป็นส่วนที่จะไว้ใช้สร้าง Model หรือให้ Model เรียนรู้

2) Testing Data เป็นส่วนที่ไว้ใช้ทดสอบประสิทธิภาพของ Model

โดยจะแบ่งเป็น Training Data 75 เปอร์เซ็นต์ และ Testing Data 25 เปอร์เซ็นต์ ในการแบ่งข้อมูลนี้ หากแบ่งให้จำนวนข้อมูล Testing Data มีน้อย จะทำให้การ Train ไม่อาจวัดผลได้ในทางปฏิบัติจริง หรือถ้าหากจำนวนข้อมูล Training Data มีน้อย จะส่งผลต่อประสิทธิภาพในการแยกแยะของ Algorithm

5. การทดลองแบบจำลอง (Fitting Model)

โดยในแต่ละวิธีมีการเรียกใช้ชุดคำสั่งในการเขียนโปรแกรมแตกต่างกัน

1. SVM : `from sklearn import svm`
2. KNN : `from sklearn.neighbors import NearestNeighbors`
3. NCC : `from sklearn.neighbors import NearestCentroid`
4. ANN : `from sklearn.neural_network import MLPClassifier`

6. ทดสอบประสิทธิภาพของแบบจำลองในการทำนายผลสำหรับข้อมูลในชุด Testing Data เพื่อหาอัตราความแม่นยำ (Accuracy Rates)

7. นำค่าอัตราความแม่นยำของทั้ง 4 การทดลองมาหาค่าเฉลี่ยสำหรับแต่ละวิธี

บทที่ 4

ผลการทดลอง

4.1 ผลการเก็บข้อมูล

รูปภาพเห็ดนางฟ้าภูฏานที่ได้จากการเก็บข้อมูลที่รักเห็ดฟาร์ม จังหวัดสุพรรณบุรี รวบรวมได้ทั้งหมด 79 รูป แบ่งเป็น กลุ่ม A 27 รูป, กลุ่ม B 8 รูป และกลุ่ม C รวมทั้งหมด 4 กลุ่มย่อยมี 44 รูป

4.2 ผลการวิเคราะห์

4.2.1 อัตราความแม่นยำจากการคัดแยกด้วยวิธี SVM และ NCC

จากการทดลอง ในการคัดแยกคุณภาพทั้งวิธี SVM และ NCC หากไม่ใช้กระบวนการ Pre-processing โดยมีผลอัตราความแม่นยำเท่ากับ 92.85 เปอร์เซ็นต์ และ 83.44 เปอร์เซ็นต์ ตามลำดับ และในการทำ PCA เพื่อลดมิติหรือแกนคุณลักษณะเด่นของข้อมูล จะทำการลดมิติของข้อมูลจาก 32,768 มิติ เหลือเพียง *N* มิติ โดยในการทดสอบ จะกำหนดค่า *N* เท่ากับ 7, 17 และ 27 ซึ่งผลอัตราความแม่นยำแสดงในตารางที่ 4.1 จะเห็นว่าวิธี SVM มีประสิทธิภาพสูงสุดเมื่อกำหนดให้ *N*=17 และ NCC มีประสิทธิภาพสูงสุดเมื่อ *N*=27

ตารางที่ 4.1 อัตราความแม่นยำจากการประมวลผลด้วยวิธี SVM และ NCC

Methods	Number of principal components		
	N=7	*N=17*	*N=27*
SVM	88.24%	94.80%	93.55%
NCC	67.34%	89.38%	90.63%

4.2.2 อัตราความแม่นยำจากการคัดแยกด้วยวิธี KNN

จากตารางที่ 4.2 การคัดแยกด้วยวิธี KNN มีผลอัตราความแม่นยำสูงสุดเมื่อ *N*=17 และค่า *k*=5

ตารางที่ 4.2 อัตราความแม่นยำจากการประมวลผลด้วยวิธี KNN

k	Without Pre-processing	Number of principal components		
		N=7	*N=17*	*N=27*
3	89.48%	88.24%	88.10%	89.48%
5	91.98%	88.96%	93.23%	90.42%
7	83.63%	81.06%	79.81%	85.01%
9	81.20%	78.24%	77.79%	79.81%

4.2.3 อัตราความแม่นยำจากการคัดแยกด้วยวิธี ANN

การคัดแยกคุณภาพด้วยวิธี ANN จะมีชั้นซ่อนตัว (Hidden Layers) ทั้งหมด 2 ชั้น โดยชั้นแรกมีทั้งหมด 16 Nodes ชั้นที่สองมี 9 Nodes และฟังก์ชันกระตุ้นที่ใช้ได้แก่ Linear Function, Sigmoid Function, Hyperbolic Tangent (Tanh) Function และ Rectified Linear Unit (ReLU) Function และในชั้นสุดท้าย (Output Layer) ใช้ฟังก์ชันกระตุ้นคือ Softmax Function ซึ่งเป็นฟังก์ชันที่ถูกนำไปใช้บ่อยในงาน Classification ที่มีเอาต์พุตหลายกลุ่ม

ตารางที่ 4.3 การ Pre-processing ด้วยวิธี Scaling สามารถเพิ่มประสิทธิภาพการคัดแยกได้โดยฟังก์ชันกระตุ้น ReLU ให้อัตราความแม่นยำที่ 98.69 เปอร์เซ็นต์

ตารางที่ 4.3 อัตราความแม่นยำจากการประมวลผลด้วยวิธี ANN

Pre-processing	Activating Function			
	linear	*sigmoid*	*Tanh*	*ReLU*
Non-scaling	9.78%	58.46%	55.52%	27.66%
Scaling	93.78%	93.44%	86.34%	98.69%

ซึ่งในการทดลอง การคัดแยกผิดประเภทเกิดขึ้นในกรณีที่คัดแยกกลุ่ม B ถุงที่มีเชื้อราเจริญเติบโตจนเกือบถุงเต็มถุง เป็นกลุ่ม A และในก้อนเพาะเชื้อเห็ดกลุ่ม C นั้นสามารถคัดแยกได้อย่างแม่นยำ

บทที่ 5

สรุปผล

5.1 สรุปผลการดำเนินงาน

งานวิจัยนี้นำเสนอการสร้างแบบจำลองที่เกิดจากการเรียนรู้ของเครื่อง (Machine Learning) เพื่อใช้ในการคัดแยกคุณภาพก้อนเพาะเชื้อเห็ดนางฟ้าภูฏาน โดยวิธี Artificial Neural Network (ANN) ใช้เทคนิคการ Scaling ข้อมูลก่อนนำไปประมวลผลนั้นมีความเหมาะสมกับการนำไปใช้งาน เนื่องจากมีอัตราความแม่นยำสูงที่สุด และสามารถคัดแยกคุณภาพก้อนเพาะเชื้อเห็ดนางฟ้าภูฏานได้อย่างมีประสิทธิภาพ โดยค่าเปอร์เซ็นต์ความผิดพลาด 1.31 เปอร์เซ็นต์ ที่เกิดขึ้นสามารถยอมรับได้เนื่องจากเกิดในกรณีที่ระบุก้อนเพาะเชื้อเห็ดกลุ่ม B ที่มีเชื้อราเจริญเติบโตเกือบเต็มถุงใกล้เคียงกับกลุ่ม A

5.2 อุปสรรคในการดำเนินงาน

1. การเก็บรักษาก้อนเพาะเชื้อเห็ดที่นำมาทดลองทำได้ยาก เนื่องจากต้องเก็บในสถานที่ที่มีการควบคุมความชื้น

2. การควบคุมแสงในงาน Image Processing ในขั้นตอนการถ่ายภาพก้อนเพาะเชื้อเห็ดนางฟ้าภูฏานควบคุมได้ยาก เนื่องจากมีการสะท้อนของถุงพลาสติก

3. การเกิดโรคระบาดของ COVID-19 ทำให้ไม่สามารถทำแบบจำลองมาสร้างเครื่องคัดกรองก้อนเพาะเชื้อเห็ดได้ตามแผนงาน

เอกสารอ้างอิง

[1] Pound. 2558. **การเพาะเห็ดนางฟ้าภูฏาน**. [ออนไลน์]. เข้าถึงได้จาก : http://farmkaset.blogspot.com/2015/07/blog-post_82.html. (สืบค้นเมื่อวันที่ : 2 พฤศจิกายน 2562).

[2] กฤษณะฟาร์มเห็ด. 2559. **เห็ดนางฟ้าภูฐาน**. [ออนไลน์]. เข้าถึงได้จาก : https://www.kritsanamushroom.com/th/galleries/26381-เห็ดนางฟ้าภูฐาน. (สืบค้นเมื่อวันที่ : 2 พฤศจิกายน 2562).

[3] Watsan Homsin. 2560. **ภาษา Python**. [ออนไลน์]. เข้าถึงได้จาก : http://marcuscode.com/lang/python. (สืบค้นเมื่อวันที่ : 19 มีนาคม 2563).

[4] Sarayut Nonsiri. 2560. **ภาษาโปรแกรม Python คืออะไร**. [ออนไลน์]. เข้าถึงได้จาก : https://www.9experttraining.com/articles/python-คืออะไร. (สืบค้นเมื่อวันที่ : 19 มีนาคม 2563).

[5] Certes. (2563). **OpenCV**. [Online]. Available : https://en.m.wikipedia.org/wiki/OpenCV. (accessed : 19 March 2020).

[6] scikit-learn developers. (2007-2019). **sklearn.decomposition.PCA**. [Online]. Available : https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html. (accessed : 22 March 2020).

[7] scikit-learn developers. (2007-2019). **sklearn.preprocessing.StandardScaler**. [Online]. Available : https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html. (accessed : 22 March 2020).

[8] ชิตพงษ์ กิตตินราดร. 2562. **Feature Scaling**. [ออนไลน์]. เข้าถึงได้จาก : https://guopai.github.io/ml-blog06.html. (สืบค้นเมื่อวันที่ : 22 มีนาคม 2563).

[9] Kan Ouivirach. 2562. **ลด Dimension ด้วย Principal Component Analysis (PCA)**. [ออนไลน์]. เข้าถึงได้จาก : https://www.digitalskill.org/contents/62. (สืบค้นเมื่อวันที่ : 22 มีนาคม 2563).

เอกสารอ้างอิง (ต่อ)

[10] Parin Kittipongdaja. 2562. **เริ่มต้นทำ Machine Learning แบบง่ายๆ**. [ออนไลน์]. เข้าถึงได้จาก : https://medium.com/@parinkittipongdaja/เริ่มต้นทำ-machine-learning-แบบง่ายๆ-อธิบายพร้อม-code-1-53a54e23e323. (สืบค้นเมื่อวันที่ : 22 มีนาคม 2563).

[11] scikit-learn developers. 2017. **Support Vector Machines**. [Online]. Available : https://scikit-learn.org/stable/modules/svm.html#classification. (accessed : 23 March 2020).

[12] scikit-learn developers. 2017. **Nearest Neighbors**. [Online]. Available : https://scikit- learn.org/stable/modules/neighbors.html#nearest-neighbors-classification. (accessed : 23 March 2020).

[13] scikit-learn developers. 2017. **Neural network models (supervised)**. [Online]. Available : https://scikit-learn.org/stable/modules/neural_networks_supervised.html#classification. (accessed : 23 March 2020).

[14] Muhammad Tayyab. 2014. **CONDUCTION OF NERVE IMPULSE**. [Online]. Available : https://simplebiologyy.blogspot.com/2014/08/conduction-of-nerve-impulse.html. (accessed : 30 March 2020).

[15] Jingjing Wang, Xiaoyu Zhang, Xiaoling Tao and Jianfeng Wang. 2018. **EPSLP: Efficient and privacy-preserving single-layer perceptron learning in cloud computing. Journal of High Speed Networks**. [Online]. Available : https://content.iospress.com/articles/journal-of-high-speed-networks/jhs594. (accessed : 4 April 2020).

[16] Facundo Bre, Juan M. Gimenez, Victor D. Fachinotti. 2017. **Prediction of wind pressure coefficients on building surfaces using Artificial Neural Networks**. [Online]. Available : https://www.researchgate.net/figure/Artificial- neural-network-architecture-ANN-i-h-1-h-2-h-n-o_fig1_321259051. (accessed : 4 April 2020).

เอกสารอ้างอิง (ต่อ)

[17] Pongthep Vijite. 2018. **ประเภทของ Machine Learning**. [ออนไลน์]. เข้าถึงได้จาก: https://medium.com/coeffest/ประเภทของ-machine-learning-f3159fee7b56. (สืบค้นเมื่อวันที่ : 13 เมษายน 2563).

[18] Pisit Bee. 2018. **Backpropagation.** [Online]. Available : https://medium.com/boobeejung/backpropagation-a0c8c6363192. (accessed : 13 April 2020).

[19] ดร.วราวุธ วุฒิวณิชย. 2561. **Artificial Neural Networks**. [ออนไลน์]. เข้าถึงได้จาก: http://irre.ku.ac.th/pubart/PubArt/39-ann.pdf. (สืบค้นเมื่อวันที่ : 13 เมษายน 2563).

[20] CS231N Staff Stanford University. 2017. **Convolutional Neural Networks for Visual Recognition**. [Online]. Available : https://cs231n.github.io/neural-networks-1/?fbclid=IwAR3ewzmiyNisuNZY8F4Q4gcy157mw7Ca3olf1RVe-tldEvhqlsBrti1AxPw. (accessed : 13 April 2020).

[21] Jason Brownlee. 2019. **A Gentle Introduction to the Rectified Linear Unit (ReLU)**. [Online]. Available : https://machinelearningmastery.com/rectified- linear-activation-function-for-deep-learning-neural-networks/. (accessed : 21 April 2020).

[22] Sagar Sharma. 2017. **Activation Functions in Neural Networks**. [Online]. Available : https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6. (accessed : 21 April 2020).

[23] Keng Surapong. 2019. **Sigmoid Function**. [Online]. Available : https://www.bualabs.com/archives/1261/what-is-activation-function-what-is-sigmoid-function-activation-function-ep-1/. (accessed : 21 April 2020).

[24] Keng Surapong. 2019. **Tanh Function**. [Online]. Available : https://www.bualabs.com/archives/1324/what-is-tanh-activation-function-compare-different-sigmoid-function-artificial-neural-network-activation-function-ep-2/. (accessed : 21 April 2020).

เอกสารอ้างอิง (ต่อ)

[25] Keng Surapong. 2019. **ReLU Function**. [Online]. Available : https://www.bualabs.com/archives/1355/what-is-relu-function-why-popular-deep-learning-training-deep-neural-network-activation-function-ep-3/. (accessed : 21 April 2020).

[26] Nagesh Singh Chauhan. 2019. **Introduction to Artificial Neural Networks (ANN)**. [Online]. Available : https://towardsdatascience.com/introduction-to-artificial-neural-networks-ann-1aea15775ef9. (accessed : 21 April 2020).

ภาคผนวก

ภาคผนวก ก
โปรแกรมการทำงานและการแสดงผล

```
from os import listdir
from os.path import isfile, join
from matplotlib import pyplot as plt
from sklearn.neighbors.nearest_centroid import NearestCentroid
from sklearn.neighbors import KNeighborsClassifier
from sklearn import svm
from sklearn import tree
from matplotlib.colors import ListedColormap
import numpy
import numpy as np
import cv2
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_validate
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.neural_network import MLPClassifier


floderCA = 'C:/Users/dell/Desktop/Class A'
filesCA = [fa for fa in listdir(floderCA) if isfile(join(floderCA,fa))]


imagesCA = numpy.empty(len(filesCA), dtype = object)
```

```
for i in range(len(filesCA)):
    #print(i)
    imagesCA[i] = cv2.imread( join(floderCA,filesCA[i]))


imgca = numpy.empty(len(filesCA), dtype = object)
for j in range(len(filesCA)):
    imgca[j] = cv2.resize(imagesCA[j],(0,0),fx=0.75,fy=0.75)


i = 0
j = 0


floderCB = 'C:/Users/dell/Desktop/Class B'
filesCB = [fb for fb in listdir(floderCB) if isfile(join(floderCB,fb))]


imagesCB = numpy.empty(len(filesCB), dtype = object)


for i in range(len(filesCB)):
    #print(i)
    imagesCB[i] = cv2.imread(join(floderCB,filesCB[i]))


imgcb = numpy.empty(len(filesCB), dtype = object)


for j in range(len(filesCB)):
```

```
    imgcb[j] = cv2.resize(imagesCB[j],(0,0),fx=0.75,fy=0.75)


i = 0

j = 0


floderCCB = 'C:/Users/dell/Desktop/Class C1'

filesCCB = [fcb for fcb in listdir(floderCCB) if isfile(join(floderCCB,fcb))]


imagesCCB = numpy.empty(len(filesCCB), dtype = object)


for i in range(len(filesCCB)):

    #print(i)

    imagesCCB[i] = cv2.imread(join(floderCCB,filesCCB[i]))


imgccb = numpy.empty(len(filesCCB), dtype = object)


for j in range(len(filesCCB)):

    imgccb[j] = cv2.resize(imagesCCB[j],(0,0),fx=0.75,fy=0.75)


i = 0

j = 0


floderCCG = 'C:/Users/dell/Desktop/Class C2'
```

```
filesCCG = [fcg for fcg in listdir(floderCCG) if isfile(join(floderCCG,fcg))]

imagesCCG = numpy.empty(len(filesCCG), dtype = object)

for i in range(len(filesCCG)):
    #print(i)
    imagesCCG[i] = cv2.imread(join(floderCCG,filesCCG[i]))

imgccg = numpy.empty(len(filesCCG), dtype = object)

for j in range(len(filesCCG)):
    imgccg[j] = cv2.resize(imagesCCG[j],(0,0),fx=0.75,fy=0.75)

i = 0
j = 0

floderCCI = 'C:/Users/dell/Desktop/Class C3'
filesCCI = [fci for fci in listdir(floderCCI) if isfile(join(floderCCI,fci))]

imagesCCI = numpy.empty(len(filesCCI), dtype = object)

for i in range(len(filesCCI)):
    #print(i)
```

```
    imagesCCI[i] = cv2.imread(join(floderCCI,filesCCI[i]))


imgcci = numpy.empty(len(filesCCI), dtype = object)


for j in range(len(filesCCI)):
    imgcci[j] = cv2.resize(imagesCCI[j],(0,0),fx=0.75,fy=0.75)


i = 0
j = 0


floderCCD = 'C:/Users/dell/Desktop/Class C4'
filesCCD = [fcd for fcd in listdir(floderCCD) if isfile(join(floderCCD,fcd))]


imagesCCD = numpy.empty(len(filesCCD), dtype = object)


for i in range(len(filesCCD)):
    #print(i)
    imagesCCD[i] = cv2.imread(join(floderCCD,filesCCD[i]))


imgccd = numpy.empty(len(filesCCD), dtype = object)


for j in range(len(filesCCD)):
        imgccd[j] = cv2.resize(imagesCCD[j],(0,0),fx=0.75,fy=0.75)
```

```
h_bins = 32

s_bins = 32

v_bins = 32

histsize = [h_bins,s_bins,v_bins]

h_range = [0,255]

s_range = [0,255]

v_range = [0,255]

hsv_range = h_range + v_range + s_range

channels = [0,1,2]


imgca_hsv = numpy.empty(len(filesCA), dtype = object)

imgcb_hsv = numpy.empty(len(filesCB), dtype = object)

imgccb_hsv = numpy.empty(len(filesCCB), dtype = object)

imgccg_hsv = numpy.empty(len(filesCCG), dtype = object)

imgcci_hsv = numpy.empty(len(filesCCI), dtype = object)

imgccd_hsv = numpy.empty(len(filesCCD), dtype = object)


histca = numpy.empty(len(filesCA), dtype = object)

histcb = numpy.empty(len(filesCB), dtype = object)

histccb = numpy.empty(len(filesCCB), dtype = object)

histccg = numpy.empty(len(filesCCG), dtype = object)

histcci = numpy.empty(len(filesCCI), dtype = object)

histccd = numpy.empty(len(filesCCD), dtype = object)
```

```
for i in range(len(filesCA)):
    imgca_hsv[i] = cv2.cvtColor(imgca[i],cv2.COLOR_BGR2RGB)
    histca[i] =
cv2.calcHist([imgca_hsv[i]],channels,None,histsize,hsv_range,accumulate=False)

i = 0

for i in range(len(filesCB)):
    imgcb_hsv[i] = cv2.cvtColor(imgcb[i],cv2.COLOR_BGR2RGB)
    histcb[i] =
cv2.calcHist([imgcb_hsv[i]],channels,None,histsize,hsv_range,accumulate=False)

i = 0

for i in range(len(filesCCB)):
    imgccb_hsv[i] = cv2.cvtColor(imgccb[i],cv2.COLOR_BGR2RGB)
    histccb[i] =
cv2.calcHist([imgccb_hsv[i]],channels,None,histsize,hsv_range,accumulate=False)

i = 0

for i in range(len(filesCCG)):
    imgccg_hsv[i] = cv2.cvtColor(imgccg[i],cv2.COLOR_BGR2RGB)
```

```
    histccg[i] =
cv2.calcHist([imgccg_hsv[i]],channels,None,histsize,hsv_range,accumulate=False)


i = 0

for i in range(len(filesCCI)):
    imgcci_hsv[i] = cv2.cvtColor(imgcci[i],cv2.COLOR_BGR2RGB)
    histcci[i] =
cv2.calcHist([imgcci_hsv[i]],channels,None,histsize,hsv_range,accumulate=False)


i = 0

for i in range(len(filesCCD)):
    imgccd_hsv[i] = cv2.cvtColor(imgccd[i],cv2.COLOR_BGR2RGB)
    histccd[i] =
cv2.calcHist([imgccd_hsv[i]],channels,None,histsize,hsv_range,accumulate=False)


c1 = 0
c2 = 0
c3 = 0
c4 = 0
c5 = 0
c6 = 0
```

```
for c1 in range(len(filesCA)):
    histca[c1] = np.reshape(histca[c1], 32768)
for c2 in range(len(filesCB)):
    histcb[c2] = np.reshape(histcb[c2], 32768)
for c3 in range(len(filesCCB)):
    histccb[c3] = np.reshape(histccb[c3], 32768)
for c4 in range(len(filesCCG)):
    histccg[c4] = np.reshape(histccg[c4], 32768)
for c5 in range(len(filesCCI)):
    histcci[c5] = np.reshape(histcci[c5], 32768)
for c6 in range(len(filesCCD)):
    histccd[c6] = np.reshape(histccd[c6], 32768)

i = 0
for i in range(len(filesCA)):
    histca[i] = histca[i].tolist()
i = 0
for i in range(len(filesCB)):
    histcb[i] = histcb[i].tolist()
i = 0
for i in range(len(filesCCB)):
    histccb[i] = histccb[i].tolist()
i = 0
```

```
for i in range(len(filesCCG)):
    histccg[i] = histccg[i].tolist()
i = 0
for i in range(len(filesCCI)):
    histcci[i] = histcci[i].tolist()
i = 0
for i in range(len(filesCCD)):
    histccd[i] = histccd[i].tolist()


#print(len(filesCA))
#print(len(filesCB))
#print(len(filesCC))


X =
np.array([histca[0],histca[1],histca[2],histca[3],histca[4],histca[5],histca[6],histca[7],histca[8],h
istca[9],histca[10],histca[11],histca[12],histca[13],

histca[14],histca[15],histca[16],histca[17],histca[18],histca[19],histca[20],histca[21],histca[22]
,histca[23],histca[24],histca[25],
        histcb[0],histcb[1],histcb[2],histcb[3],histcb[4],histcb[5],histcb[6],

histccb[0],histccb[1],histccb[2],histccb[3],histccb[4],histccb[5],histccb[6],histccb[7],histccb[8
],

histccb[9],histccb[10],histccb[11],histccb[12],histccb[13],histccb[14],histccb[15],histccb[16],
histccb[17],histccb[18],histccb[19],histccb[20],histccb[21],
```

```
histccg[0],histccg[1],histccg[2],histccg[3],histccg[4],histccg[5],histccg[6],histccg[7],histccg[8],h
istccg[9],histccg[10],histccg[11],

        histcci[0],histcci[1],histcci[2],histcci[3],

        histccd[0],histccd[1],histccd[2],histccd[3],histccd[4],histccd[5],histccd[6],histccd[7]])

y = np.array(['Class A','Class A','Class A','Class A','Class A','Class A','Class A','Class A','Class
A','Class A','Class A','Class A','Class A','Class A','Class A','Class A',

        'Class A','Class A','Class A','Class A','Class A','Class A','Class A','Class A','Class
A','Class A',

        'Class B','Class B','Class B','Class B','Class B','Class B','Class B',

        'Class C(Black Fungus)','Class C(Black Fungus)',

        'Class C(Black Fungus)','Class C(Black Fungus)','Class C(Black Fungus)','Class
C(Black Fungus)','Class C(Black Fungus)','Class C(Black Fungus)','Class C(Black Fungus)',

        'Class C(Black Fungus)',

        'Class C(Black Fungus)','Class C(Black Fungus)','Class C(Black Fungus)',

        'Class C(Black Fungus)','Class C(Black Fungus)','Class C(Black Fungus)',

        'Class C(Black Fungus)','Class C(Black Fungus)','Class C(Black Fungus)',

        'Class C(Black Fungus)','Class C(Black Fungus)',

        'Class C(Black Fungus)',

        'Class C(Green Fungus)','Class C(Green Fungus)','Class C(Green Fungus)','Class
C(Green Fungus)','Class C(Green Fungus)','Class C(Green Fungus)','Class C(Green Fungus)',

        'Class C(Green Fungus)',

        'Class C(Green Fungus)',

        'Class C(Green Fungus)',

        'Class C(Green Fungus)',
```

```
                'Class C(Green Fungus)',
                'Class C(Insects)','Class C(Insects)',
                'Class C(Insects)',
                'Class C(Insects)',
                'Class C(Mushroom Substrate is Died)',
                'Class C(Mushroom Substrate is Died)',
                'Class C(Mushroom Substrate is Died)',
                'Class C(Mushroom Substrate is Died)',
                'Class C(Mushroom Substrate is Died)',
                'Class C(Mushroom Substrate is Died)',
                'Class C(Mushroom Substrate is Died)','Class C(Mushroom Substrate is Died)'])

sc = StandardScaler()
X = sc.fit_transform(X)
pca = PCA(n_components=17)
X = pca.fit_transform(X)


#num1 = 0
#num2 = []
#print("BG Removed")
#print("PCA Variance Ratio")
#print(pca.explained_variance_ratio_)
#for i in pca.explained_variance_ratio_:
```

```
#    num2.append(i + num1)
#    num1 = i + num1


#test = cv2.imread('C:/Users/dell/Desktop/Test/CCCC1.PNG')
#test = cv2.resize(test, (0,0), fx=0.75, fy=0.75)
#test = cv2.cvtColor(test, cv2.COLOR_BGR2RGB)
#histtest = cv2.calcHist([test], channels, None, histsize, hsv_range, accumulate=False)
#histtest = np.reshape(histtest, 32768)
#histtest = histtest.tolist()
#histtest = sc.transform([histtest])
#histtest = pca.transform(histtest)


clf = MLPClassifier(solver = 'lbfgs', alpha = 1e-5, hidden_layer_sizes = (64, 64, 64),
random_state = 1)
clf.fit(X, y)
#clf = NearestCentroid(metric = 'euclidean')
#clf.fit(X, y)
#scores = cross_val_score(clf,X,y,cv=4)
#print()
#print('Cross Validation of Nearest Centroid Method (shrink threshold=0)')
#print(scores)
#print()
#print('K = ',i+1)
#clf = KNeighborsClassifier(n_neighbors = 1)
```

```
#clf.fit(X, y)

#print('Cross Validation of Knn Method (k=5)')

#print(scores)

#print()

#clf = svm.SVC(gamma = 'scale', kernel = 'sigmoid')

#clf.fit(X,y)

#print(clf.predict(histtest))

#clf = tree.DecisionTreeClassifier(max_depth = 5)

#clf = clf.fit(X, y)

#cv_results = cross_validate(clf, X, y, cv = 4)

#sorted(cv_results.keys())

#dep = 0

#for i in cv_results['score_time']:

#    dep += i

#print(dep/4)


accuracy = cross_val_score(clf, X, y, cv = 4)

precision = cross_val_score(clf, X, y, cv = 4, scoring = 'precision_weighted')

recall = cross_val_score(clf, X, y, cv = 4, scoring = 'recall_weighted')

f1 = cross_val_score(clf, X, y, cv = 4, scoring = 'f1_weighted')


#box1 = 0

#box2 = 0
```

```
#box3 = 0

#box4 = 0


#print('ANN Classifier (Relu)')

#print('Accuracy')

#for i in accuracy:

#    box1 += i

#print(box1*100/4)

#print('Precision')

#for i in precision:

#    box2 += i

#print(box2*100/4)

#print('Recall')

#for i in recall:

#    box3 += i

#print(box3*100/4)

#print('F1 Score')

#for i in f1:

#    box4 += i

#print(box4*100/4)

#print('Average = Weighted')

#print()

#precision = cross_val_score(clf, X, y, cv = 4, scoring = 'precision_micro')
```

```
#recall = cross_val_score(clf, X, y, cv = 4, scoring = 'recall_micro')
#f1 = cross_val_score(clf, X, y, cv = 4, scoring = 'f1_micro')
print('Accuracy')
print(accuracy)
print('Precision')
print(precision)
print('Recall')
print(recall)
print('F1 Score')
print(f1)
print('Average = Weighted')
#print()
#precision = cross_val_score(clf, X, y, cv = 4, scoring = 'precision_macro')
#recall = cross_val_score(clf, X, y, cv = 4, scoring = 'recall_macro')
#f1 = cross_val_score(clf, X, y, cv = 4, scoring = 'f1_macro')
#print('Precision')
#print(precision)
#print('Recall')
#print(recall)
#print('F1 Score')
#print(f1)
#print('Average = Macro')
```