การวิเคราะห์และทำนายคุณภาพน้ำด้วยหลักการเรียนรู้ของเครื่อง

Water Quality Analysis and Prediction Using Machine Learning

พีระพัฒน์ เดชาธิติรัฐ

Peerapat Dechathitirat

ภาณุวิชญ์ ภู่มี

Phanuwich Phumee

จุฬาลักษณ์ ล้อมวงศ์ไพบูลย์

Juralux Lomwongpaibool

โครงงานนี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรวิทยาศาสตรบัณฑิต

ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์

มหาวิทยาลัยศรีนครินทรวิโรฒ ปีการศึกษา 2562

การวิเคราะห์และทำนายคุณภาพน้ำด้วยหลักการเรียนรู้ของเครื่อง

Water Quality Analysis and Prediction Using Machine Learning

พีระพัฒน์ เดชาธิติรัฐ

Peerapat Dechathitirat

ภาณุวิชญ์ ภู่มี

Phanuwich Phumee

จุฬาลักษณ์ ล้อมวงศ์ไพบูลย์

Juralux Lomwongpaibool

โครงงานนี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรวิทยาศาสตรบัณฑิต

ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์

มหาวิทยาลัยศรีนครินทรวิโรฒ ปีการศึกษา 2562

คณะวิทยาศาสตร์ มหาวิทยาลัยศรีนครินทรวิโรฒ

ชื่อหัวข้อโครงงาน	**การวิเคราะห์และทำนายคุณภาพน้ำด้วยหลักการเรียนรู้ของเครื่อง** **Water Quality Analysis and Prediction Using Machine Learning**		
นิสิต	นายพีระพัฒน์	เดชาธิติรัฐ	59102010272
	นายภาณุวิชญ์	ภู่มี	59102010274
	นางสาวจุฬาลักษณ์	ล้อมวงศ์ไพบูลย์	59102010899
ปริญญา	วิทยาศาสตรบัณฑิต (วท.บ.)		
ภาควิชา	วิทยาการคอมพิวเตอร์		
อาจารย์ที่ปรึกษาโครงงาน	ผู้ช่วยศาสตราจารย์ ดร.ศุภชัย ไทยเจริญ		

ลงชื่อ.....

(ผู้ช่วยศาสตราจารย์ ดร.ศุภชัย ไทยเจริญ)

บทคัดย่อ

น้ำเป็นทรัพยากรที่สำคัญในการดำรงชีวิตของมนุษย์และสิ่งมีชีวิต ในปัจจุบันมีการขยายตัวของเศรษฐกิจและสังคมอย่างมาก ส่งผลให้สภาพแวดล้อมต่างๆเสื่อมโทรมลงรวมถึงแหล่งน้ำด้วย ซึ่งแหล่งน้ำจำเป็นจะต้องได้รับการตรวจสอบคุณภาพน้ำอยู่เสมอ โดยแนวทางในการตรวจสอบคุณภาพน้ำจะต้องทำการตรวจสอบและวิเคราะห์ผ่านห้องทดลองทางวิทยาศาสตร์ ซึ่งใช้เวลาอย่างมากจึงไม่สามารถทำให้วางแผนและแก้ไขสถานการณ์คุณภาพน้ำได้อย่างทันที จุดประสงค์ของงานวิจัยนี้คือการพัฒนาการตรวจสอบคุณภาพน้ำโดยการวิเคราะห์และจำแนกประเภทคุณภาพน้ำออกเป็น 3 ประเภท ได้แก่ ดี, พอใช้ และเสื่อมโทรม โดยใช้ข้อมูลคุณภาพน้ำ 4 พารามิเตอร์ ได้แก่ อุณหภูมิ, กรด-เบส, ความขุ่น และออกซิเจนที่ละลายในน้ำ ด้วย machine learning นำแบบจำลองหลากหลายประเภทมาทำการทดลอง เพื่อหาแบบจำลองที่เหมาะกับ ชุดข้อมูลคุณ ภาพน้ำในประเทศไทย และ ทำการปรับไฮเปอร์พารามิเตอร์ (hyperparameter) ของแต่ละแบบจำลองเพื่อเพิ่มประสิทธิภาพของแบบจำลอง จากผลทดลองพบว่าแบบจำลองโครงข่ายประสาทเทียม (Neural Network) มีประสิทธิภาพดีที่สุดในการทำนายผลของคุณภาพน้ำ โดยให้ค่าความถูกต้อง (Accuracy) เท่ากับ 84%

Abstract

Water is one of the most important natural resources for all creatures. It plays a vital role in almost every aspects of life particularly for human beings. Unfortunately, with the expansion of economy and society, many water sources have been severely deteriorated in such a way that they could pose a health risk to people. As a result, the quality of water for consumption should be examined regularly. However, water quality is typically determined through experiments and analyses in a laboratory, which requires time, resource, and expertise. To address this issue, this paper presents a simple but effective water quality analysis and classification technique that does not rigorous investigation and lab work. The method is based on machine-learning algorithms and four physical water parameters, temperature, acid-base, turbidity, and dissolved oxygen. To prepare training/test datasets, raw water datasets from Pollution Control Department, Thailand are put through some pre-processing tasks such as Q-value normalization and standard WQI computation. Water quality are divided into three class labels, good, fair, and deteriorated. A handful of selected machine-learning algorithms are used for building classification models from the data. According to the experimental results, the best water quality classification model is constructed by a Neural Network using with the accuracy of 84%.

กิตติกรรมประกาศ

การตรวจสอบคุณภาพคุณภาพน้ำโดยการใช้หลักการเรียนรู้ของเครื่อง (Machine learning) ผู้จัดทำได้มุ่งมั่นศึกษาและค้นคว้าอย่างต่อเนื่องจนได้รับความรู้และความเข้าใจจนสามารถพัฒนาเป็นการตรวจสอบคุณภาพน้ำอย่างรวดเร็วได้ โครงงานนี้สามารถสำเร็จลุล่วงเพราะได้รับความเมตตาและความช่วยเหลือจากบุคคลหลายๆ ท่านตั้งแต่เริ่มต้นโครงงานจนสำเร็จเสร็จสิ้น คณะผู้จัดทำขอขอบพระคุณบุคคลทุกท่านที่ให้ความรู้ คำแนะนำ แนวทางการแก้ไขปัญหา และให้ความอนุเคราะห์เพื่อนการศึกษาโครงงานนี้

ขอขอบพระคุณสำนักจัดการคุณภาพน้ำ กรมควบคุมมลพิษ ที่ให้ความอนุเคราะห์ข้อมูลคุณภาพน้ำอันเป็นประโยชน์ต่อการจัดทำโครงงานในครั้งนี้

ขอขอบพระคุณ ผศ.ดร.ศุภชัย ไทยเจริญ ซึ่งเป็นอาจารย์ที่ปรึกษาโครงงานนี้ซึ่งท่านได้ให้คำแนะนำและข้อคิดเห็นต่างๆ อันเป็นประโยชน์อย่างยิ่งในการทำวิจัย ตลอดจนปรับปรุงแก้ไขข้อบกพร่องต่างๆ ที่เกิดขึ้นระหว่างการดำเนินการทำโครงงาน รวมทั้งให้กำลังแก่คณะผู้จัดทำเสมอมา

ขอขอบพระคุณอาจารย์ทุกท่านในภาควิชาวิทยาการคอมพิวเตอร์ มหาวิทยาลัยศรีนครินทรวิโรฒ ที่กรุณาให้คำแนะนำ ข้อคิดเห็น และความรู้ต่างๆ ที่เป็นประโยชน์ต่อการจัดทำโครงงานในครั้งนี้

ขอขอบคุณเพื่อนๆนิสิตหลักสูตรวิทยาศาสตรบัณฑิต ภาควิชาวิทยาการคอมพิวเตอร์ มหาวิทยาลัยศรีนครินทรวิโรฒ ที่ช่วยให้คำแนะนำอันเป็นประโยชน์และช่วยเหลือเกื้อกูลเพื่อให้โครงงานนี้ลุล่วงไปได้ด้วยดี

สุดท้ายนี้ขอขอบพระคุณครอบครัวของผู้จัดทำที่ให้การช่วยเหลือและสนับสนุนมาโดยตลอด รวมทั้งให้ความห่วงใยและเป็นกำลังใจที่สำคัญในการทำโครงงานเสมอมา

นายพีระพัฒน์ เดชาธิติรัฐ

นายภาณุวิชญ์ ภู่มี

นางสาวจุฬาลักษณ์ ล้อมวงศ์ไพบูลย์

สารบัญ

สารบัญรูปภาพ

สารบัญตาราง

บทที่ 1 บทนำ

1.1 ที่มาและความสำคัญ

โลกของเราประกอบด้วยพื้นน้ำประมาณ 3 ส่วน (75%) และพื้นดิน 1 ส่วน (25%) ซึ่งเป็นปัจจัยที่สำคัญในการดำรงชีวิตของมนุษย์และสิ่งมีชีวิตต่างๆ รวมการใช้ประโยชน์ของน้ำในด้านต่างๆ ทั้งในด้านการอุปโภคบริโภค ด้านเกษตรกรรมและอุตสาหกรรม เนื่องจากในปัจจุบันการเติบโตและการขยายตัวทางเศรษฐกิจและสังคมอย่างกว้างขวางส่งผลให้แหล่งน้ำหลายแห่งแย่ลง และยังพบว่าอัตราน้ำเสียในพื้นที่กรุงเทพมหานครมากกว่า 80% มาจากแหล่งชุมชน ซึ่งเกิดจากการใช้น้ำในชีวิตประจำวันของพวกเราทุกคน โดยโรงงานควบคุมคุณภาพน้ำส่วนใหญ่มีขีดความสามารถในการบำบัดน้ำเสีย 1,112,000 ลูกบาศก์เมตรต่อวัน ซึ่งระบบบำบัดน้ำเสียที่มีอยู่สามารถบำบัดเพียง 45% ของปริมาณน้ำเสียทั้งหมดต่อวัน [1] ซึ่งทำให้มีน้ำเสียจำนวนมากที่ยังไม่ได้ผ่านการบำบัด และน้ำที่มนุษย์ใช้ดำรงชีวิตและบริโภคจะต้องสะอาด บริสุทธิ์ ปราศจากเชื้อโรคและสารพิษปนเปื้อน ดังนั้นเราจึงจำเป็นจะต้องตรวจสอบคุณภาพแหล่งน้ำอย่างสม่ำเสมอ

โดยคณะผู้จัดทำเล็งเห็นถึงความสำคัญของการตรวจสอบคุณภาพน้ำ เนื่องจากการตรวจสอบคุณภาพน้ำในอดีตจะทำการตรวจสอบผ่านห้องทดลองทางวิทยาศาสตร์จึงใช้ระยะเวลานานในการตรวจสอบ จึงได้ทำการศึกษาหลักการการเรียนรู้ของเครื่อง (Machine learning) โดยการนำข้อมูลคุณภาพน้ำ 4 ค่า ได้แก่ ค่าอุณหภูมิ, ค่าความเป็นกรด-ด่าง, ค่าความขุ่น, ค่าออกซิเจนละลายในน้ำ มาวิเคราะห์และสร้างแบบจำลองทำนายคุณภาพน้ำและสามารถที่จะทำให้นักวิชาการ นักวิจัย หรือผู้ที่มีหน้าที่รับผิดชอบในส่วนงานนั้นๆ สามารถที่จะวางแผนและเตรียมพร้อมรับมือกับปัญหาที่จะเกิดได้ล่วงหน้าและรวดเร็ว

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 เพื่อศึกษาหลักการในการตรวจสอบคุณภาพน้ำ

1.2.2 เพื่อศึกษาหลักการวิทยาการข้อมูลที่เกี่ยวข้องกับการวัดคุณภาพน้ำ

1.2.3 เพื่อประยุกต์ใช้หลักการวิเคราะห์ข้อมูลและการเรียนรู้ของเครื่องในการทำนายคุณภาพน้ำ

1.3 ขอบเขตของโครงงาน

1.3.1 ใช้ชุดข้อมูล (Data set) จากสำนักจัดการคุณภาพน้ำ กรมควบคุมมลพิษซึ่งเป็นข้อมูลคุณภาพน้ำของประเทศไทย

1.3.2 นำข้อมูลคุณสมบัติของน้ำที่สามารถตรวจวัดได้ด้วยอุปกรณ์เซ็นเซอร์มาวิเคราะห์กับแบบจำลองการเรียนรู้

1.3.3 ใช้แบบจำลอง 5 แบบจำลอง ได้แก่ Gaussian Naive Bayes, Random Forest, Neural Network : Multilayer Perceptron (MLP), K-Nearest Neighbor และ Logistic Regression ในการทำนายคุณภาพน้ำ

1.3.4 ใช้ค่า precision, recall, accuracy และ f1-score ในการวัดประสิทธิภาพของแบบจำลอง

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1.4.1 ได้แบบจำลองสำหรับการวิเคราะห์และทำนายคุณภาพน้ำที่ไม่ซับซ้อนและมีประสิทธิภาพ

1.4.2 สามารถนำแบบจำลองที่ได้ไปพัฒนาเพื่อให้สามารถวางแผนและแก้ไขสถานการณ์น้ำเสียได้อย่างทันที

1.4.3 นิสิตได้ความรู้ทางด้านการวัดคุณภาพน้ำ หลักการการเรียนรู้ของเครื่อง รวมถึงทักษะประสบการณ์ในการทำงานร่วมกันเป็นทีม

บทที่ 2 เอกสารและงานวิจัยที่เกี่ยวข้อง

2.1 งานวิจัยที่เกี่ยวข้อง

2.1.1 ชื่อเรื่อง : Smart Data Analysis for Water Quality in Catchment Area Monitoring [2]

ชื่อผู้แต่ง : Di Wu,Hadi Mohammed,Hao Wang,Razak Seidu

บทความนี้นำเสนอการวิเคราะห์และประมาณการณ์คุณภาพน้ำ โดยเก็บข้อมูลจากแหล่งน้ำในประเทศนอร์เวย์ เมื่อได้ข้อมูลแล้วจึงนำไปวิเคราะห์ตัวชี้วัดคุณภาพน้ำทางชีวภาพโดยใช้แบบจำลอง Zero-Inflated Poisson Regression (ZIP) และ Zero-inflated Negative Binomial Regression (ZINB) เพื่อเปรียบเทียบปัจจัยที่มีผลที่แตกต่างกัน และแสดงให้เห็นถึงความแม่นยำของแบบจำลองที่ใช้ในการวิเคราะห์และประมาณการณ์คุณภาพน้ำในอนาคต

2.1.2 ชื่อเรื่อง : IoT Based Real-time River Water Quality Monitoring System [3]

ชื่อผู้แต่ง : Mohammad Salah Uddin Chowdury, Talha Bin Emran,Subhasish Ghosh,Abhijit Pathak,Mohd. Manjur Alam,Nurul Absar,Mohammad Shahadat Hossain

บทความนี้นำเสนอการสร้างแบบจำลองการทำนายคุณภาพน้ำเสียแบบเรียลไทม์ โดยใช้ Neural Network และเก็บข้อมูล จากสถานที่ห่างไกลและนำมาเพื่อตรวจสอบคุณภาพ และสามารถแจ้งเตือนหรือแก้ไขคุณภาพของน้ำได้อย่างทันท่วงที

2.1.3 ชื่อเรื่อง : Analysis and Prediction of Water Quality Using LSTM Deep Neural Networks in IoT Environment [4]

ชื่อผู้แต่ง : Ping Liu, Jin Wang,Arun Kumar Sangaiah,Yang Xie,Xinchun Yin

บทความนี้นำเสนอการสร้างแบบจำลองการทำนายค่าพารามิเตอร์คุณภาพน้ำ ในบทความวิจัยนี้ได้มีการนำตัวแปรที่เกี่ยวข้องกับเวลามาช่วยในการวิเคราะห์ เพื่อสะท้อนความผันผวนของคุณภาพน้ำในช่วงเวลาต่างๆที่จุดตรวจสอบ โดยแบบจำลองที่ใช้คือ LSTM Deep Neural Networks

2.1.4 ชื่อเรื่อง : Water Quality Monitoring Using Wireless Sensor Networks : Current Trends and Future Research Directions [5]

ชื่อผู้แต่ง : Kofi Sarpong Adu-Manu, Cristiano Tapparello, Wendi Heinzelman, Ferdinand Apietu Katsriku, Jamal-Deen Abdulai

เป็นบทความวิจัยที่รวบรวมงานวิจัยต่าง ๆ ที่เกี่ยวข้องกับการทำระบบตรวจสอบคุณภาพของน้ำ โดยช่วงเวลาของบทความวิจัยอยู่ในช่วงปี 2002-2013

2.1.5 ชื่อเรื่อง : Efficient Water Quality Prediction Using Supervised Machine Learning [6]

ชื่อผู้แต่ง : Umair Ahmed, Rafia Mumtaz, Hirra Anwar, Asad A. Shah, Rabia Irfan, José García-Nieto

เป็นบทความที่กล่าวถึงวิธีการทำนายค่า water quality index (WQI) และ water quality class (WQC) รวมถึงวิธีการเตรียมข้อมูล การอธิบายเกี่ยวกับแบบจำลองต่างๆ ในการทำนายค่าและสรุปผลประสิทธิภาพของแต่ละแบบจำลอง

2.1.6 ชื่อเรื่อง : Water Quality Indices [7]

ชื่อผู้แต่ง : A.K. Thukral, Renu Bhardwaj, Rupinder Kaur

บทความนี้กล่าวถึงการคำนวณดัชนีคุณภาพน้ำ (Water Quality Index : WQI) ค่าน้ำหนักของแต่ละพารามิเตอร์ในการคำนวณดัชนีคุณภาพน้ำ รวมถึงการแปลงค่าพารามิเตอร์ให้เป็น Q-Value ที่อยู่ในช่วง 0 ถึง 100

2.2 ทฤษฎีและแนวคิดที่ใช้ในการทำโครงงาน

2.2.1 องค์ความรู้เกี่ยวกับดัชนีคุณภาพน้ำ (Water Quality Index : WQI)

ดัชนีคุณภาพน้ำ (Water Quality Index : WQI) เป็นสิ่งที่บ่งบอกถึงระดับคุณภาพน้ำว่ามีคุณภาพอยู่ในช่วงใด ซึ่งจะบ่งชี้ว่าแหล่งน้ำดังกล่าวจะต้องดำเนินการควบคุมและดูแลอย่างไร ดัชนีคุณภาพน้ำนี้ถูกพัฒนาโดย National Sanitation Foundation (NSF) ประเทศสหรัฐอเมริกา ในปี 1970 ดัชนีคุณภาพน้ำดังกล่าวคำนึงถึงพารามิเตอร์คุณสมบัติของน้ำ 9 พารามิเตอร์ ดังนี้

1. Dissolved oxygen (mg/L)
2. Fecal coliform (number/100 ml)
3. pH (standard units)

4. Biochemical oxygen demand – 5 day (mg/L)

5. Temperature change (1 mile, 0C)

6. Total phosphate – P (mg/L)

7. Nitrates (mg/L)

8. Turbidity (NTU)

9. Total solids (mg/L)

โดยการคำนวนค่า WQI นั้นจำเป็นต้องแปลงค่าแต่ละพารามิเตอร์ให้อยู่ในอัตราส่วนจาก 0 ถึง 100 ซึ่งจะเรียกค่าที่อยู่ในอัตราส่วนนั้นว่าค่า Q-Value หลังจากนั้นจะทำการนำค่า Q-Value แต่ละพารามิเตอร์ที่ได้ไปเข้าสูตรคำนวณดัชนีคุณภาพน้ำ โดยมีค่าน้ำหนักของพารามิเตอร์ที่แตกต่างกันออกไป

$$WQI = \frac{\sum(Q - Value \times Weighting\ factor)}{\sum Weighting\ factors}$$

ค่าน้ำหนักพารามิเตอร์

พารามิเตอร์	ค่าน้ำหนัก
Dissolved oxygen	0.17
Fecal coliform	0.16
pH	0.11
BOD	0.11
Temperature	0.10
Total Phosphate	0.10
Nitrate	0.10
Turbidity	0.08
Total Solids	0.07

ตารางที่ 1 : ค่าน้ำหนักของแต่ละพารามิเตอร์

หลังจากที่ได้ค่าดัชนีคุณภาพน้ำ(WQI) จะทำการนำค่าดัชนีคุณภาพน้ำไปจะจัดอันดับเป็น 5 อันดับ ดังนี้

ค่าดัชนีคุณภาพน้ำ	ระดับคุณภาพน้ำ
>90-100	ดีมาก
>70-90	ดี
>50-70	พอใช้
>25-50	เสื่อมโทรม
0-25	เสื่อมโทรมมาก

ตารางที่ 2 : เกณฑ์คุณภาพน้ำ

2.2.2 องค์ความรู้เกี่ยวกับ Data Science [8]

เกี่ยวข้องกับการศึกษาและการสร้างอัลกอริทึมที่สามารถเรียนรู้ข้อมูลและทำนายข้อมูลได้ อัลกอริทึมนั้นจะทำงานโดยอาศัยแบบจำลองที่สร้างมาจากชุดข้อมูลตัวอย่างขาเข้าเพื่อการทำนายหรือตัดสินใจในภายหลัง แทนที่จะทำงานตามลำดับของคำสั่งโปรแกรมคอมพิวเตอร์ การเรียนรู้ของเครื่องมีเกี่ยวข้องอย่างมากกับสถิติศาสตร์ นอกจากนี้ยังมีความสัมพันธ์กับสาขาการหาค่าเหมาะที่สุดในทางคณิตศาสตร์ที่แง่ของวิธีการ ทฤษฎี และการประยุกต์ใช้ การเรียนรู้ของเครื่องสามารถนำไปประยุกต์ใช้งานได้หลากหลาย

การเรียนรู้ของเครื่อง สามารถแบ่งโดยกว้างๆได้เป็น 3 ประเภท ตามประเภทของ "ข้อมูลฝึก" หรือ "ข้อมูลขาเข้า" ได้ดังนี้

1. Supervised Learning คือ การเรียนรู้แบบมีผู้สอน ข้อมูลตัวอย่างและผลลัพธ์ที่ "ผู้สอน" ต้องการถูกป้อนเข้าสู่คอมพิวเตอร์ แล้วให้คอมพิวเตอร์ทำการสร้างกฎทั่วไปที่สามารถเชื่อมโยงข้อมูลขาเข้ากับขาออกได้

2. Unsuperviesd Learning คือ การสร้างแบบจำลองที่เหมาะสมกับข้อมูล จะไม่มีการระบุผลที่ต้องการหรือประเภทไว้ก่อน การเรียนรู้แบบนี้จะพิจารณาวัตถุเป็นเซตของตัวแปรสุ่ม แล้วจึงสร้างแบบจำลองความหนาแน่นร่วมของชุดข้อมูล

3. Reinforcement Learning คอมพิวเตอร์ทำการเรียนรู้และปฏิสัมพันธ์กับสิ่งแวดล้อมที่เปลี่ยนไปตลอดเวลาโดยคอมพิวเตอร์จะต้องทำงานบางอย่าง เช่น ขับรถโดยที่ไม่มีผู้สอนคอยบอกอย่างจริงจังว่าวิธีการที่ทำอยู่นั้นเข้าใกล้เป้าหมายแล้วหรือไม่ ตัวอย่างเช่น การเรียนรู้เพื่อเล่นเกม

2.2.3 องค์ความรู้เกี่ยวกับ Exploratory data analysis (EDA) [9]

Exploratory data analysis (EDA) เป็นวิธีการวิเคราะห์ชุดข้อมูลเพื่อสรุปลักษณะสำคัญของข้อมูล เป็นกระบวนการที่สำคัญในการดำเนินการตรวจสอบข้อมูลเบื้องต้นเพื่อหารูปแบบ ค้นหาจุดผิดปกติ วิธีการทำ EDA มีดังนี้

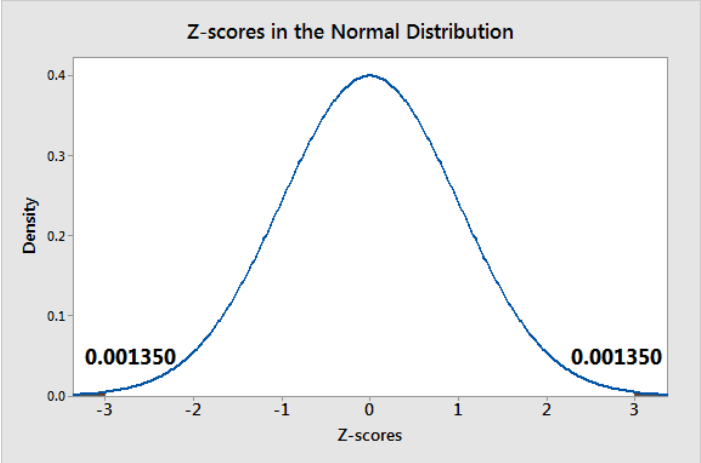
การตรวจสอบค่าสูญหาย (Missing Values)

ในสถิติข้อมูลที่หายไปหรือค่าที่หายไปเกิดขึ้นเมื่อไม่มีการเก็บค่าข้อมูลสำหรับตัวแปร ข้อมูลที่หายไปเป็นเหตุการณ์ที่เกิดขึ้นทั่วไปและอาจมีผลกระทบอย่างมีนัยสำคัญ ต่อข้อสรุปที่ดึงออกมาจากข้อมูล

การตรวจสอบค่าผิดปกติ (Outlier/Extreme)

Outlier คือในทางสถิติเป็นจุดข้อมูลที่แตกต่างกันอย่างมีนัยสำคัญจากข้อสังเกตอื่นๆ ค่าผิดปกติอาจเกิดจากความแปรปรวนของการวัดหรืออาจแสดงข้อผิดพลาดจากการทดลอง บางครั้งจะถูกแยกออกจากชุดข้อมูล ค่าผิดปกติอาจจะให้เกิดข้อมูลผิดพลาดในการวิเคราะห์ได้ โดยวิธีจัดการข้อมูลที่ผิดพลาดมีหลายวิธีหนึ่งในวิธีการจัดการนั้นก็คือ Z-score Outlier

โดย Z-score สามารถหาปริมาณความผิดปกติของข้อมูลได้เมื่อข้อมูลของคุณ เป็นไปตามการแจกแจงแบบปกติ Z-score คือจำนวนของส่วนเบี่ยงเบนมาตรฐานด้านบนและต่ำกว่าค่าเฉลี่ยที่แต่ละค่าตก ยิ่งคะแนน Z ของค่านั้นไกลออกไปเท่าไหร่ยิ่งผิดปกติมากเท่านั้น ค่าเกณฑ์มาตรฐานสำหรับการค้นหาค่าผิดปกติคือ Z-score ที่ +/- 3 หรือมากกว่าจากศูนย์



รูปที่ 1 : Z-score ในการแจกแจงแบบปกติ

(https://statisticsbyjim.com/basics/outliers/)

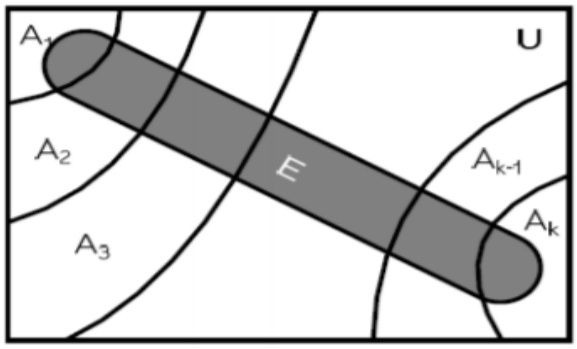
การเตรียมข้อมูล (Data preparation)

กระบวนการเมื่อมีชุดข้อมูลที่ไม่พร้อมสำหรับการประมวลผลโดยอัลกอริทึมการเรียนรู้ของเครื่อง ในบางกรณีเมื่อข้อมูลมีความซับซ้อน จะทำให้เป็นข้อมูลเป็นไปในบรรทัดฐานของข้อมูลเดียวกัน แจกจ่ายข้อมูลให้เท่ากันในช่วงที่ยอมรับได้สำหรับอัลกอริทึมการเรียนรู้ของเครื่อง

2.2.4 องค์ความรู้เกี่ยวกับ Gaussian Naive Bayes [10]

ทฤษฎีบทของเบย์ (Bayes' Theorem) เป็นสูตรทางคณิตศาสตร์อย่างง่ายที่ใช้สำหรับการคำนวณความน่าจะเป็นตามเงื่อนไขมาพัฒนาเป็นทฤษฎีบทดังกล่าว โดยมีรายละเอียดดังต่อไปนี้

ถ้าให้เอกภพสัมพัทธ์ U ประกอบด้วยเหตุการณ์ที่ไม่สามารถเกิดขึ้นได้พร้อมกัน จำนวน k เหตุการณ์คือ A1, A2, A3, ..., Ak ดังรูป และให้ E เป็นเหตุการณ์หนึ่งในปริภูมิตัวอย่างที่เกิดจากการทดลองเดียวกันนี้และต้องเป็นส่วนหนึ่งของ Ai (i = 1, 2, 3, ..., k)



รูปที่ 2 : เหตุการณ์ E บนเหตุการณ์ k เหตุการณ์ที่เกิดพร้อมกันไม่ได้

(ที่มา : https://mis.csit.sci.tsu.ac.th/noppamas/download/DataMining/DataMiningCh7V1.pdf)

Naive Bayes Classifier (การเรียนรู้แบบเบย์)

การเรียนรู้แบบเบย์ เป็นเทคนิคที่ใช้ทฤษฎีความน่าจะเป็นตามกฎของเบย์ (Bayes' Theorem) เพื่อหาว่าสมมติฐานใดน่าจะถูกต้องที่สุด โดยใช้ความรู้ก่อนหน้า (Prior Knowledge) ได้แก่ ความน่าจะเป็นก่อนหน้าสำหรับสมมติฐานหนึ่งๆ ร่วมกับข้อมูล เช่น ความน่าจะเป็นที่สังเกตได้สำหรับสมมติฐานหนึ่ง ๆ เพื่อหาสมมติฐานที่ดีที่สุด

การเรียนรู้แบบเบย์อาศัยหลักการของการคำนวณความน่าจะเป็นของแต่ละสมมติฐาน (คลาสเป้าหมายหรือผลลัพธ์การทำนาย) โดยการเรียนรู้แบบเบย์เป็นการเรียนรู้เพิ่มได้ เนื่องจากตัวอย่างใหม่ที่ได้มาถูกนำมาปรับเปลี่ยนการแจกแจงซึ่งมีผลต่อการเพิ่ม หรือ ลดความน่าจะเป็น ทำให้มีการเรียนรู้ที่เปลี่ยนไป วิธีการนี้ตัวแบบจะถูกปรับเปลี่ยนไปตามตัวอย่างใหม่ที่ได้โดยผนวกกับความรู้เดิมที่มี ซึ่งการทำนายค่าคลาสเป้าหมายของตัวอย่างใช้ความน่าจะเป็นมากที่สุดของทุกสมมตฐาน จากทฤษฎีของเบย์ จะสามารถคำนวณความน่าจะเป็นของสมมติฐานต่าง ๆ โดยใช้สมการต่อไปนี้

$$P(c|x) = \frac{P(x|c) * P(c)}{P(x)}$$

x แทนข้อมูลที่นำมาใช้ในการคำนวณการแจกแจงความน่าจะเป็น posteriori probability ของสมมติฐาน c คือ P (c | x) ตามทฤษฎี

P (c) คือ ความน่าจะเป็นก่อนหน้าของสมมติฐาน c

P (x) คือ ความน่าจะเป็นก่อนหน้าของชุดข้อมูลตัวอย่าง x

P (c | x) คือ ความน่าจะเป็นของ c เมื่อรู้ x

P(x | c) คือ ความน่าจะเป็นของ x เมื่อรู้ c

Gaussian Naive Bayes [11]

Naive Bayes สามารถที่จะประยุกต์กับข้อมูลที่เป็นจำนวนจริง (continuous variables) ได้ โดยทั่วไปแล้วจะอาศัยหลักการการกระจายแบบเกาส์เซียน (Gaussian distribution) ทำให้การประยุกต์ใช้ของ Naïve Bayes นี้เรียกว่า Gaussian Naive Bayes

การประยุกต์ของ Naive Bayes กับข้อมูลที่เป็นจำนวนจริง โดยทั่วไปจะใช้ฟังก์ชัน ในการประมาณการกระจายตัวของข้อมูล แต่ Gaussian (หรือการแจกแจงแบบปกติ) เป็นวิธีที่ง่ายที่สุดในการทำงานเพราะเพียงแค่ประมาณค่าเฉลี่ยและค่าเบี่ยงเบนมาตรฐานจากชุดข้อมูล (Train data-set)

หลักการของ Gaussian Naive Bayes คือจะต้องคำนวณความน่าจะเป็นสำหรับค่าอินพุตสำหรับแต่ละคลาสโดยใช้ความถี่ เนื่องด้วยข้อมูลป้อนเข้าที่เป็นจำนวนจริง จะสามารถคำนวณค่าเฉลี่ยและส่วนเบี่ยงเบนมาตรฐานของค่าอินพุต(x)สำหรับแต่ละคลาสเพื่อสรุปการแจกแจงได้ หมายความว่านอกจากที่จะต้องเก็บความน่าจะเป็นของแต่ละคลาสแล้ว จะต้องเก็บค่าเฉลี่ยและส่วนเบี่ยงเบนมาตรฐานสำหรับแต่ละคลาสด้วย

การทำนายโดยใช้แบบจำลอง Gaussian Naive Bayes คือความน่าจะเป็นของค่าป้อนเข้าใหม่ (x_i) จะถูกคำนวณโดยใช้ Gaussian Probability Density Function (PDF) โดยมีสมการดังต่อไปนี้

$$P(x_i|c) = \frac{1}{\sqrt{2 * \pi * \sigma_{x_i,c}^2}} * \exp\left(-\frac{(x_i - \mu_{x_i,c})^2}{2 * \sigma_{x_i,c}^2}\right)$$

โดยหากการทำนายเกิดจากการใช้ข้อมูลที่มีคุณสมบัติหลายตัว ก็จะต้องทำการคำนวณหา Gaussian Probability Density Function (PDF) ของแต่ละตัว จากนั้นจึงนำมาทำการคูณกัน จากนั้นเมื่อคำนวณค่าความน่าจะเป็นของค่าป้อนเข้าใหม่เรียบร้อยแล้ว ความน่าจะเป็นแบบมีเงื่อนไข (Conditional probability) ของแต่ละคลาส จะถูกคำนวณโดยใช้ทฤษฎีบทของเบย์

$$P(c) = \frac{Number\ of\ instances\ of\ class\ c}{Total\ number\ of\ instances\ in\ dataset}$$

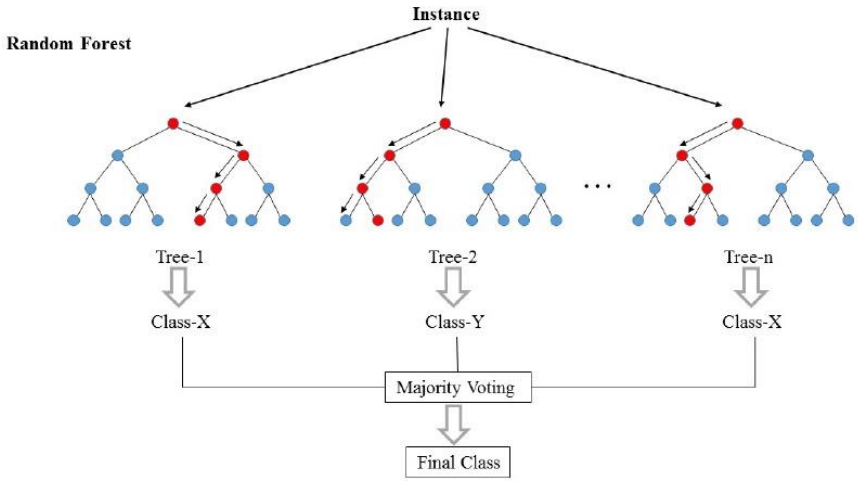
$$P(c_i|x) = \frac{P(x|c_i) * P(c_i)}{\sum_j P(x|c_j) * P(c_j)}$$

ค่า $P(c_i|x)$ จะถูกคำนวณซ้ำตามคลาสที่มีทั้งหมด และคลาสที่แสดงความน่าจะเป็นสูงสุดก็จะเป็นคำตอบของการทำนาย

2.2.5 องค์ความรู้เกี่ยวกับ Random Forest [12]

หลักการของ Random Forest คือ การสร้างแบบจำลองจาก Decision Tree หลายๆ แบบจำลองย่อยๆ (ตั้งแต่ 10 แบบจำลองถึงมากกว่า 1,000 แบบจำลอง) โดยแต่ละแบบจำลองจะได้รับข้อมูลที่แตกต่างกันซึ่งเป็นส่วนย่อยของข้อมูลทั้งหมด เวลาทำการทำนายผลลัพธ์ก็จะทำการนำผลที่ได้แต่ละแบบจำลองมารวมกันพร้อมนับจำนวนผลที่มีจำนวนซ้ำกันมากที่สุด และนำออกมาเป็นผลลัพธ์สุดท้าย Decision Tree มีลักษณะคล้ายกับโครงสร้างของต้นไม้ โดยที่ภายในต้นไม้จะประกอบไปด้วย

โหนด (Node) ซึ่งแต่ละโหนดนั้นจะแสดงถึงการตัดสินใจ บนข้อมูลของคุณสมบัติต่างๆ กิ่งของต้นไม้จะแสดงถึงค่าหรือผลลัพธ์ที่ได้จากการทดสอบ



รูปที่ 3 : Random Forest

(ที่มา : http://www.nrronline.org/viewimage.asp?img=NeuralRegenRes_2018_13_6_962_233433_f2.jpg)

Hyperparameter ของ Random Forest

n_estimators คือ จำนวนต้นไม้ที่จะใช้ในการทำ Random Forest การเลือกจำนวนมากๆ ใน Random forest ไม่ได้ช่วยทำให้ประสิทธิภาพของแบบจำลองนั้นดีขึ้น แม้ว่ามันจะไม่ทำให้ประสิทธิภาพมันน้อยลง แต่ก็สามารถช่วยทำให้เกิดความซับซ้อนในการคำนวณ

max_depth คือ กำหนดความยาวของเส้นทางตั้งแต่โหนดรากไปจนถึงโหนดใบ ถ้าเพิ่มจำนวนของ max depth จะทำให้ประสิทธิภาพ ของ training set เพิ่มขึ้นเรื่อยๆ แต่ในทางตรงกันข้าม test set ประสิทธิภาพจะเพิ่มขึ้นถึงจุดนึงและหลังจากประสิทธิภาพจะค่อยๆ ลดลง

min_samples_split คือ จำนวนขั้นต่ำที่จำเป็นในการแบ่งโหนดภายใน หาก int ให้พิจารณา min_samples_split เป็นตัวเลขขั้นต่ำ และหากเป็น float min_samples_split จะพิจารณาเป็นเศษส่วน

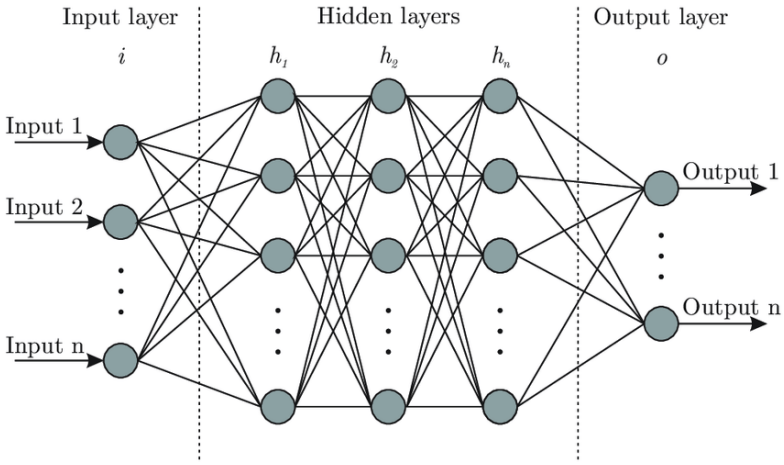
min_samples_leaf คือ ระบุจำนวนข้อมูลขั้นต่ำที่ควรมีในโหนดใบหลังจากการแยกโหนดหากมีจำานวนข้อมูลต่ำากว่า min_samples_leaf ให้หยุดการแบ่งโหนดนั้นๆเพื่อเป็นการลดโอกาส overfitting โดยจะกำหนดค่า min_samples_leaf ให้สอดคล้องตามขนาดของชุดข้อมูล

max_features คือ (0.0–1.0) โดยระบุว่าแต่ละ decision tree ใน Random Forest จะสามารถสุ่มหยิบ feature ได้มากที่สุดกี่เปอร์เซ็นต์ (0–100%) ซึ่งการไม่กำหนดค่าดังกล่าวสูงจนเกินไปจะเป็นการลดความสัมพันธ์กันเองของ tree (ลด correlation) และลดโอกาสการเกิด overfit ของแบบจำลอง

2.2.6 องค์ความรู้เกี่ยวกับ Neural Network : Multilayer Perceptron (MLP) [13]

Neural Network (NN) หรือโครงข่ายประสาทเทียม คือ แบบจำลองทางคณิตศาสตร์หรือแบบจำลองทางคอมพิวเตอร์สำหรับประมวลผลสารสนเทศด้วยการคำนวณแบบคอนเนคชันนิสต์ (connectionist) แนวคิดเริ่มต้นของเทคนิคนี้ได้มาจากการศึกษาโครงข่ายไฟฟ้าชีวภาพ (bioelectric network) ในสมอง ซึ่งประกอบด้วย เซลล์ประสาท (neurons) และ จุดประสานประสาท (synapses) ตามแบบจำลองนี้ ข่ายงานประสาทเกิดจากการเชื่อมต่อระหว่างเซลล์ประสาทจนเป็นเครือข่ายที่ทำงานร่วมกัน

ส่วนประกอบของ Neural Network



รูปที่ 4 : ส่วนประกอบของ Neural Network

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

1. Neurons (โหนด) ข้างใน Neuron จะต่างกันตาม layer ที่มันอยู่ โดยถ้าเป็น Input layer จะมีข้อมูลที่รับมา แต่ถ้าเป็น Hidden layer ก็จะมีสมการที่ช่วยในการคำนวณเพื่อการทำงานและส่งค่าไปยัง layer ต่อไป แต่ถ้าเป็น Output ก็จะเป็นตัวที่บ่งชี้ถึงคำตอบ

2. Input Layer มีหน้าที่ในการรับข้อมูลเข้ามาในโครงข่ายประสาทโดย Input layer จะเพียงชั้นเดียวเท่านั้นและมีหน้าส่งข้อมูลไปยังชั้นถัดไป (Hidden layer)

3. Hidden Layer มีหน้าที่รับข้อมูลจาก layer ก่อนหน้า layer นี้ สามารถมีจำนวนมากกว่า 1 และโดยพื้นฐาน ถ้ายิ่งต้องการความแม่นยำที่มากขึ้นก็จะสามารถเพิ่มจำนวนชั้นของ Hidden layer และจำนวน Neurons ให้มากขึ้นได้แต่ก็ไม่เสมอไปในกรณีอื่น ๆ

4. Output Layer เป็นชั้นสุดท้ายที่รอรับค่าจาก hidden layer ชั้นสุดท้าย

โครงข่ายประสาทเทียมแบบ MLP เป็นรูปแบบหนึ่งของโครงข่ายประสาทเทียมที่มีโครงสร้างเป็นแบบหลายๆ ชั้น ใช้สำหรับงานที่มีความซับซ้อนได้ผลเป็นอย่างดี โดยมีกระบวนการฝึกฝนเป็นแบบ Supervised Learning และใช้ขั้นตอนการส่งค่าย้อนกลับ (Backpropagation) สำหรับการฝึกฝน กระบวนการส่งค่าย้อนกลับ ประกอบด้วย 2 ส่วนย่อยคือ การส่งผ่านไปข้างหน้า (Forward Pass) การส่งผ่านย้อนกลับ (Backward Pass) สำหรับการส่งผ่านไปข้างหน้า ข้อมูลจะผ่านเข้าโครงข่ายประสาทเทียมที่ชั้นข้อมูลเข้า (Input Layer) และจะส่งผ่านจากอีกชั้นหนึ่ง (Hidden Layer) ไปสู่อีกชั้นหนึ่งจนกระทั่งถึงชั้นข้อมูลออก (Output layer) ส่วนการส่งผ่านย้อนกลับค่าน้ำหนักการเชื่อมต่อแต่ละ Neuron จะถูกปรับเปลี่ยนให้สอดคล้องกับกฎการแก้ข้อผิดพลาด (Error-Correction) คือผลต่างของผลตอบที่แท้จริง (Actual Response) กับผลตอบเป้าหมาย (Target Response) เกิดเป็นสัญญาณผิดพลาด (Error Signal) ซึ่งสัญญาณผิดพลาดนี้จะถูกส่งย้อนกลับเข้าสู่โครงข่ายประสาทเทียม และค่าน้ำหนักของการเชื่อมต่อในแต่ละ Neuron จะถูกปรับจนกระทั่งผลตอบที่แท้จริงเข้าใกล้ผลตอบเป้าหมายสัญญาณที่มีโครงข่ายประสาทเทียมแบบ MLP มี 2 ประเภทคือ Function Signal และ Error Signal

1. Function Signal เป็นสัญญาณเข้าที่มาจากโหนดในชั้นก่อนหน้า และจะส่งผ่านไปข้างหน้าจากโหนดหนึ่งไปสู่อีกโหนดหนึ่ง

2. Error Signal เป็นสัญญาณย้อนกลับที่เกิดขึ้นที่โหนดในชั้นข้อมูลออกของโครงข่ายประสาทเทียม และถูกส่งผ่านย้อนกลับจากชั้นหนึ่งไปสู่อีกชั้นหนึ่ง

หลักการทำงานของ MLP คือในแต่ละโหนดของชั้น Hidden จะมีฟังก์ชันสำหรับคำนวณเมื่อได้รับสัญญาณ (Output) จากโหนดในชั้นก่อนหน้านี้ เรียกว่า Activation Function โดยในแต่ละชั้นไม่จำเป็นต้องเป็นฟังก์ชันเดียวกันก็ได้ Hidden layer นั้นมีหน้าที่สำคัญคือ จะพยายามแปลงข้อมูลที่เข้ามาในชั้นนั้น ๆ ให้สามารถแยกแยะความแตกต่างโดยใช้เส้นตรงเส้นเดียว (Linearly Separable) และก่อนที่ข้อมูลจะถูกส่งไปถึงชั้นข้อมูลออก (Output Layer) ในบางครั้งอาจจำเป็นต้องใช้ชั้น Hidden มากกว่า 1 ชั้นในการแปลงข้อมูลให้อยู่ในรูป Linearly Separable

ในการคำนวณหา Output ในปัญหาการจำแนกทำได้โดยการใส่ข้อมูล Input เข้าไปในโครงข่ายประสาทเทียม จากนั้นให้ทำการเปรียบเทียบค่าของ Output ใน Output Layer และให้ทำการเลือกค่าของ Output ที่มีค่าสูงกว่า (Neuron ที่มีค่าสูงกว่า) และทำการรับค่าของพยากรณ์ที่ตรงกับ Neuron ที่เลือกและให้นำค่าของ มาเปรียบเทียบกับค่าที่ยอมรับได้ หากค่าของอยู่ในช่วงที่รับได้ (Error น้อยกว่า Error ที่เรากำหนด) ก็ให้ทำการรับข้อมูลชุดถัดไป แต่หากค่าของมากกว่าค่าที่ยอมรับได้ ให้ทำการปรับค่าน้ำหนักและ Biased ตามขั้นตอนที่ได้กล่าวไว้ข้างต้น เมื่อทำการปรับน้ำหนักเรียบร้อยแล้ว ให้ทำการรับข้อมูลชุดถัดไปและทำตามขั้นตอนซ้ำอีกรอ จนกระทั่งถึงข้อมูลชุดสุดท้าย และเมื่อทำข้อมูลชุดสุดท้ายเสร็จจะนับเป็น 1 รอบของการคำนวณ (1 Epoch) จากนั้นจะทำการหาค่าผิดพลาดรวมเฉลี่ยจากค่าเฉลี่ยของที่ได้เก็บค่าเอาไว้ เพื่อใช้ในการตรวจสอบว่าค่าโดยเฉลี่ยในการจำแนกนั้น มีค่าน้อยกว่าค่าผิดพลาดที่ยอมรับได้หรือไม่ ถ้าใช่แสดงว่าโครงข่ายประสาทเทียมที่สร้างขึ้นนั้นสามารถให้ผลลัพธ์ที่ถูกต้องของทุก ๆข้อมูลแล้ว จึงทำการจบการเรียนรู้ได้ แต่ถ้าไม่ใช่ ให้กลับไปทำตามขั้นตอนแรก โดยเริ่มรับข้อมูลชุดที่ 1 ใหม่

2.2.7 องค์ความรู้เกี่ยวกับ K-Nearest Neighbor [14]

การจำแนกประเภทข้อมูลด้วยวิธีการค้นหาเพื่อนบ้านใกล้สุด (k-Nearest-Neighbor Classifiers) เป็นการเรียนรู้โดยการเปรียบเทียบกันระหว่างข้อมูลที่ต้องการจำแนก/ทำนายหมวดหมู่กับข้อมูลอื่นๆ ทั้งหมดในชุดข้อมูลสอน (Training set) ที่มีลักษณะเหมือนกันหรือใกล้เคียงกันด้วยการพิจารณาข้อมูลคุณสมบัติ (Attribute) ต่าง ๆ โดยข้อมูลหนึ่ง ๆจะสามารถถูกมองว่าเป็นจุดหนึ่งในระนาบ n มิติ (เมื่อ n คือจำนวนคุณสบัติทั้งหมด) ถ้านำข้อมูลทุกๆ จุดใน Training set มาวางในระนาบ n มิติ จากนั้นนำข้อมูลที่ต้องการจำแนก/ทำนายหมวดหมู่มาวางในระนาบด้วยเช่นกัน จากนั้นพิจารณาหาว่ามีข้อมูลจุดใดบ้าง ที่มีคุณลักษณะใกล้เคียงกับข้อมูลที่ต้องการจำแนกหมวดหมู่มากที่สุดเป็นจำนวน k เรคคอร์ด

ในการที่จะหาความเหมือนกัน ต่างกัน หรือค่าความใกล้เคียงกันระหว่างข้อมูลใด ๆ จะสามารถประยุกต์ใช้มาตรวัดความแตกต่างต่าง ๆ เช่น Euclidean distance ซึ่งจะทำการพิจารณาข้อมูล 2 ข้อมูล จากนั้นทำการพิจารณาความแตกต่างระหว่างค่าของแต่ละคุณสมบัติ ระหว่าง 2 ข้อมูลแล้วนำมารวมเป็นค่าความแตกต่างรวม ซึ่งจะสามารถคำนวณได้ดังนี้

$$distance(x_1, x_2) = \sqrt{\sum_{i=1}^{n}(x_{1i} - x_{2i})^2}$$

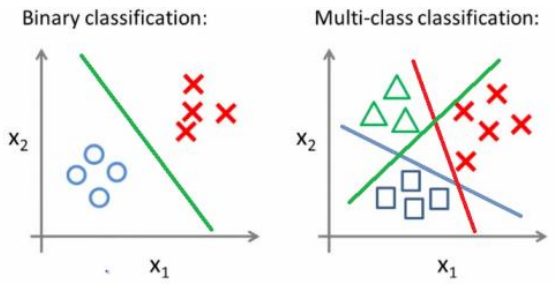
หลังจากทำการหาค่าความแตกต่างระหว่างข้อมูลที่ต้องการจำแนกหมวดหมู่กับข้อมูลทั้งหมดใน Training set แล้ว ต่อไปจะทำการจำแนก/ทำนายหมวดหมู่ของข้อมูล ด้วยการพิจารณาจากหมวดหมู่ของข้อมูล k ข้อมูลที่มีค่าความแตกต่างน้อยที่สุด k อันดับแรก (เรียกว่าเพื่อนบ้านใกล้สุด k อันดับ, k-nearest neighbor) เทียบกับข้อมูลที่ต้องการจำแนกหมวดหมู่ด้วยการค้นหาว่าใน k ข้อมูล มีหมวดหมู่ข้อมูลใดมากที่สุด จากนั้นจึงจะสรุปว่าข้อมูลที่ต้องการจำแนก/ทำนายหมวดหมู่ มีหมวดหมู่ตามหมวดหมู่ที่มากที่สุดที่พิจารณาไปข้างต้น

2.2.8 องค์ความรู้เกี่ยวกับ Logistic Regression [15]

การถดถอยแบบโลจิสติก (Logistic Regression) เป็น Machine Learning ประเภท Classification Model โดยแบบจำลองจะทำนายออกมาเป็นข้อมูลเชิงคุณภาพ (Qualitative Variable) เช่น สีผิว อาชีพ เชื้อชาติ โดยการถดถอยแบบโลจิสติกผลลัพธ์ที่ได้ออกมานั้นจะมีค่าอยู่ในช่วงระหว่าง 0 กับ 1 เนื่องจากเป็นฟังก์ชัน sigmoid โดยจะใช้ค่าทั้งสองสูงสุดและต่ำสุดเสมอ โดยการคำนวณฟังก์ชัน sigmoid ของ X โดย X คือผลรวมถ่วงน้ำหนักของคุณสมบัติของตัวแปรต้น และหลังจากนั้นก็จะได้ผลลัพธ์ความน่าจะเป็นออกมาที่จะอยู่ในช่วง 0 กับ 1

$$Sigmoid(x) = \frac{1}{1 + e^{-x}}$$

การถดถอยแบบโลจิสติกเป็นเพียงตัวจำแนกแบบไบนารี่ ซึ่งหมายความว่าพวกเขาไม่สามารถจัดการผลลัพธ์ที่มีมากกว่าสองคลาส แต่อย่างไรก็ตามการถดถอยโลจิสติกมีส่วนขยายสำหรับการถดถอยโลจิสติกส์เพื่อจำแนกแบบหลายประเภทที่เรียกว่า One VS Rest Logistic Regression (OVR) เป็นวิธีการ Heuristic สำหรับการใช้การจำแนกแบบไบนารี่เพื่อการจำแนกแบบหลายประเภท โดยการแบ่งชุดข้อมูลแบบหลายประเภท ออกเป็นเป็นหลายๆปัญหาการจำแนกไบนารี่ โดยการจำแนกแบบไบนารี่ในแต่ละปัญหาจะได้ทำการเรียนรู้ โดยคาดการณ์จากแบบจำลองที่มีความถูกต้องมากที่สุด



รูปที่ 5 : Multiclass Classification

(ที่มา : https://utkuufuk.com/2018/06/03/one-vs-all-classification)

Hyperparameter ของ Logistic Regression

penalty {‘l1’, ‘l2’, ‘elasticnet’, ‘none’}, default=’l2’ ใช้เพื่อระบุบรรทัดฐานที่ใช้ในการลงโทษ (Penalization)

Tol คือ ความอดทน(Tolerance) สำหรับเกณฑ์การหยุด (default=1e-4)

Solver คือ การเลือกอัลกอริทึมที่ใช้ในการ optimization problem (default = ‘lbfgs’)

- ‘Liblinear’ เหมาะสำหรับชุดข้อมูลที่มีขนาดเล็ก
- ‘Sag’, ’newton-cg’, ’saga’, ’lbfgs’ ใช้สำหรับจัดการปัญหาที่มีการจำแนกหลายประเภท
- ‘newton-cg’, ‘lbfgs’, ‘sag’ and ‘saga’ สามารถใช้กับ Penalty ที่เป็น L2 เท่านั้น
- ‘liblinear’ and ‘saga’ สามารถใช้กับ Penalty ที่เป็น L1 เท่านั้น
- ‘saga’ also supports สามารถใช้กับ Penalty ที่เป็น elasticnet เท่านั้น
- ‘liblinear’ ไม่สามารถใช้กับ Penalty ได้

Max_iter คือ จำนวนการวนซ้ำสูงสุดที่ solver ต้องมาบรรจบกัน

C คือ Regularization ใช้ penalty เพื่อเพิ่มขนาดของค่าพารามิเตอร์เพื่อลดการ overfitting (default = 1.0)

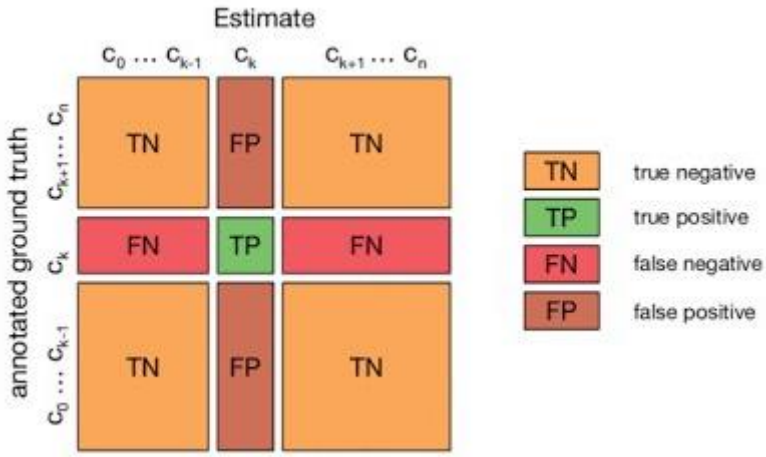
multi_class {‘auto’, ‘ovr’, ‘multinomial’} การเลือกวิธีการแยก label default=‘auto’

2.2.9 องค์ความรู้เกี่ยวกับ Synthetic Minority Over-Sampling Technique (SMOTE) [16]

Synthetic Minority Over-sampling Technique : SMOTE เป็นเทคนิคที่ใช้ในการแก้ปัญหาที่ต้องการจำแนกข้อมูลไม่สมดุล ซึ่งข้อมูลมีจำนวนตัวอย่างแตกต่างกันมากในแต่ละกลุ่ม เมื่อทำการจำแนกประเภท จะทำให้มีการเรียนรู้แต่ข้อมูลกลุ่มที่มาก ผลที่ได้ก็จะจำแนกไปในข้อมูลกลุ่มมากวิธี SMOTE เป็นวิธีการเพิ่มจำนวนข้อมูลประเภทที่มีข้อมูลน้อย ให้เพิ่มปริมาณข้อมูลใกล้เคียงกับประเภทที่มีมากที่สุดโดยสุ่มค่าขึ้นมาหนึ่งค่าและหาค่าระยะห่างระหว่างค่าที่เลือกกับทุกๆ ค่า เลือกค่าที่ใกล้เคียงที่สุด

2.2.10 องค์ความรู้เกี่ยวกับ Confusion Matrix [17]

Confusion Matrix เป็นการจำลองรูปแบบการอธิบายประสิทธิภาพของแบบจำลองและเป็นเครื่องมือที่สำคัญในการประเมินผลลัพธ์ในการทำนายของแบบจำลอง



รูปที่ 6 : Confusion Matrix

(ที่มา : https://devopedia.org/confusion-matrix)

จากรูปที่ 6 Actual Values คือ ค่าข้อมูลที่เป็นคำตอบสำหรับการทำนาย ส่วน Predicted Values คือค่าผลการทำนายของแบบจำลอง โดยค่าในตำแหน่งต่างๆ ของ Matrix จะเป็นดังนี้

True Positive (TP) คือ สิ่งที่ทำนายตรงกับสิ่งที่เกิดขึ้นจริง กล่าวคือการทำนายสิ่งที่สนใจได้ถูกต้อง

False Positive (FP) คือ สิ่งที่ทำนายไม่ตรงกับสิ่งที่เกิดขึ้น กล่าวคือการทำนายสิ่งที่ไม่สนใจว่าเป็นสิ่งที่สนใจ

False Negative (FN) คือ สิ่งที่ทำนายไม่ตรงกับที่เกิดขึ้นจริง กล่าวคือการทำนายว่าสิ่งที่สนใจเป็นสิ่งที่ไม่สนใจ

True Negative (TN) คือ สิ่งที่ทำนายตรงกับสิ่งที่เกิดขึ้น กล่าวคือการทำนายสิ่งที่ไม่สนใจได้ถูกต้อง

Precision คือความแม่นยำ มีสูตรในการคำนวณคือ Precision = TP / (TP + FP) กล่าวคือเป็นอัตราส่วนของการทำนายจำนวนกลุ่มที่สนใจได้ถูกต้อง ต่อการทำนายว่าเป็นกลุ่มสนใจทั้งหมดทั้งแบบทำนายถูกและทำนายผิด ซึ่งสรุปได้ว่าถ้าต้องการค่าความแม่นยำ เป็น 100% ก็ต้องให้ค่า FP ที่ทำนายผิดมีค่าเป็น 0

Recall คือ ความอ่อนไหว มีสูตรในการคำนวณคือ Recall = TP / (TP + FN) คือเป็นอัตราส่วนของการทำนายกลุ่มที่สนใจได้ถูกต้อง ต่อกลุ่มที่สนใจทั้งหมดที่ปรากฏในข้อมูลที่สังเกตุ ซึ่งสรุปได้ว่าถ้าความอ่อนไหว 100% ก็ต้องให้ค่า FN ที่ทายผิดมีค่าเป็น 0

Accuracy คือ ความถูกต้องโดยรวม กล่าวคือ สามารถทำนายได้ถูกต้องทั้งหมดหมดโดยไม่แยกว่าเป็นแบบ Positive และ Negative ในอัตราส่วนใด โดยดูจากสูตรแล้ว Accuracy = (TP + TN) / (TP+FP+TN+FN) สรุปได้ว่า ถ้าทายถูกทั้งหมด แล้วค่า Accuracy จะเท่ากับ 1 หรือ 100% แต่ในทางปฏิบัติจริงไม่มีทางที่จะเป็นได้ในลักษณะนี้

F1-score คือ ค่าน้ำหนักเฉลี่ยของ Recall และ Precision โดย F1-Score = 2 * (Precision * Recall / (Precision + Recall)) สรุปแบบตามแนวทางทฤษฎีสู่การปฏิบัติคือค่าที่ดีที่สุดคือ 1 และแย่สุดคือ 0

บทที่ 3 วิธีการดำเนินโครงงาน

3.1 วิธีการดำเนินงาน

3.1.1 ศึกษาหลักการและทฤษฎีที่เกี่ยวข้องกับงานวิจัย และอุปกรณ์ที่เกี่ยวข้องต่างๆ

3.1.2 จัดเตรียมและวิเคราะห์ข้อมูล

3.1.3 ออกแบบและพัฒนาแบบจำลอง

3.1.4 ทดสอบและปรับปรุงแบบจำลอง

3.1.5 สรุปผลการวิจัย เสนอแนะ และจัดทำรูปเล่ม

3.2 แผนการดำเนินงานตลอดโครงงาน

แผนดำเนินงานโครงการวิจัย								
ขั้นตอนดำเนินการ	ต.ค. 62	พ.ย. 62	ธ.ค. 62	ม.ค. 63	ก.พ. 63	มี.ค. 63	เม.ย. 63	พ.ค. 63
1. เริ่มต้นและวางแผนโครงการ								
1.1 วางแผนจัดทำโครงการ								
1.2 หาข้อมูลศึกษาแนวคิดและทฤษฎี								
1.3 ศึกษาระบบงานที่เกี่ยวข้อง								
2. การวิเคราะห์ข้อมูล								
2.1 วิเคราะห์ข้อมูลและทำ pre-processing								
3. การออกแบบแบบจำลอง								
3.1 กาหนดวัตถุประสงค์และขอบเขตของแบบจำลอง								
4. การพัฒนาแบบจำลอง								
4.1 พัฒนาแบบจำลอง								
4.2 ทดสอบและปรับปรุงแบบจำลอง								
5. สรุปและเผยแพร่งานวิจัย								
5.1 สรุปผลการดำเนินงาน								
5.2 นำเสนอผลงานที่สมบูรณ์								

ตารางที่ 3 : ตารางแผนการดำเนินงานตลอดโครงงาน

3.3 อุปกรณ์และเครื่องมือที่ใช้

3.3.1 ฮาร์ดแวร์ (Hardware)

3.3.1.1 iMac (21.5-inch, Late 2015, MacOS Mojave) 3 เครื่อง

3.3.2 ซอฟต์แวร์ (Software)

3.3.2.1 Google Colaboratory

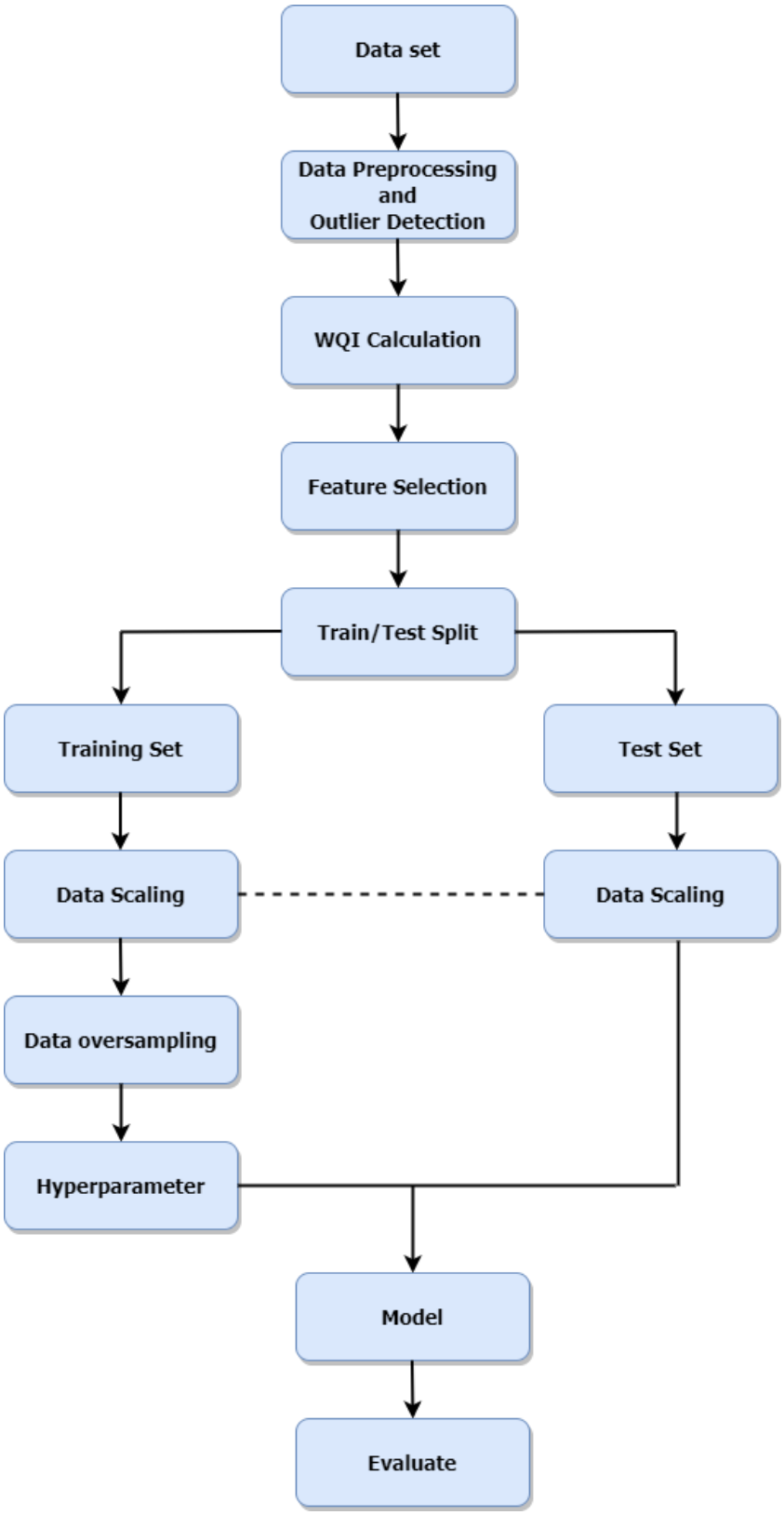
3.3.3 ภาษาที่ใช้

3.3.3.1 Python

3.4 ชุดข้อมูลที่ใช้งาน

ข้อมูลคุณภาพน้ำจากสำนักจัดการคุณภาพน้ำ กรมควบคุมมลพิษ ซึ่งประกอบด้วยข้อมูลดังนี้ (1) ชื่อแม่น้ำ, (2) ครั้งที่, (3) สถานี, (4) ปี, (5) วันที่, (6) เวลา, (7) ค่าออกซิเจนละลาย (DO), (8) ค่า Fecal Coliform Bacteria (FCB), (9) ค่า pH, (10) ค่า Biochemical Oxygen Demand (BOD), (11) ค่าอุณหภูมิน้ำ, (12) ค่าอุณหภูมิอากาศ, (13) ค่า NO3-N, (14) ค่าความขุ่น (turbidity), (15) ค่า Total Phosphate (TP), (16) ค่า Total Solids (TS), (17) ค่า Total Dissolved Solids (TDS), (18) ค่า Suspended Solids (SS), (19) ค่า NH3-N, (20) ค่า Total Coliform Bacteria (TCB) และ (21) ค่าการนำไฟฟ้า (Conductivity) จากแหล่งน้ำทุกแหล่งของทุกภาค ในช่วงวันที่ 1 มกราคม พ.ศ.2560 ถึง 31 ธันวาคม พ.ศ.2562

3.5 ขั้นตอนการทำงาน



รูปที่ 7 : แผนผังการทำงาน

3.5.1 การเตรียมข้อมูล (Data Preprocessing) : Exploratory data analysis (EDA)

ข้อมูลที่ได้จากกรมควบคุมมลพิษ มีทั้งหมด 4069 แถว 21 คอลัมน์ ลำดับแรกทำการลบคอลัมน์ที่ไม่ใช้ เช่น วันที่ เวลา สถานีที่วัด ครั้งที่และอื่นๆ ออกจนเหลือ 9 คอลัมน์ (1) ค่าอุณหภูมิน้ำ, (2) ค่า pH, (3) ค่าความขุ่น (turbidity), (4) ค่าออกซิเจนละลาย (DO), (5) ค่า Biochemical Oxygen Demand (BOD), (6) ค่า Fecal Coliform Bacteria (FCB), (7) ค่า Total Phosphate (TP), (8) ค่า NO3-N และ (9) ค่า Total Solids (TS) จากข้อมูลทั้งหมดที่กล่าว มีข้อมูลในบางคอลัมน์และแถวที่ค่าหายไป (Missing Values) รวมกันทุกคอลัมน์ 1317 แถว ทำการลบแถวที่หายไปทั้งหมดจนเหลือข้อมูล 2752 แถว สาเหตุที่ต้องลบค่าที่หายไปทั้งหมดโดยที่ไม่ทำการแก้ปัญหาแบบอื่น เช่น นำค่าเฉลี่ยมาเติม เพราะว่าจำเป็นที่จะต้องนำข้อมูลคุณสมบัติน้ำไปคำนวณค่าดัชนีคุณภาพน้ำ ทำให้ไม่เหมาะสมที่จะนำค่าที่ไม่ใช่ค่าจริงมาคำนวณ

	Temp	pH	Tur	DO	BOD	FCB	TP	NO3-N	TS
0	29.5	7.8	20.7	2.5	1.8	3500.0	0.24	0.73	176.5
1	29.8	7.6	14.0	3.9	2.7	220.0	0.31	0.81	239.9
2	29.3	7.6	16.8	4.9	2.7	700.0	0.22	0.74	208.0
3	29.3	7.4	16.3	4.8	1.4	2400.0	0.14	0.07	233.6
4	29.0	7.1	13.1	2.3	1.9	16000.0	0.14	0.60	194.4
...	...	...	...	...	...	...	...	...	...
2747	28.3	5.9	109.0	6.2	1.3	24000.0	0.39	0.62	1406.0
2748	27.9	6.6	19.0	6.0	0.7	220.0	0.41	0.38	633.0
2749	28.0	6.6	45.0	6.2	1.5	2400.0	0.41	0.41	91.0
2750	29.7	6.4	19.0	7.6	0.7	1300.0	0.45	0.37	135.0
2751	27.6	6.7	28.0	7.6	0.8	790.0	0.54	0.29	162.0

2752 rows × 9 columns

รูปที่ 8 : ตารางข้อมูลจากกรมควบคุมมลพิษ

3.5.2 การคำนวณค่าดัชนีคุณภาพน้ำ (Water Quality Index: WQI)

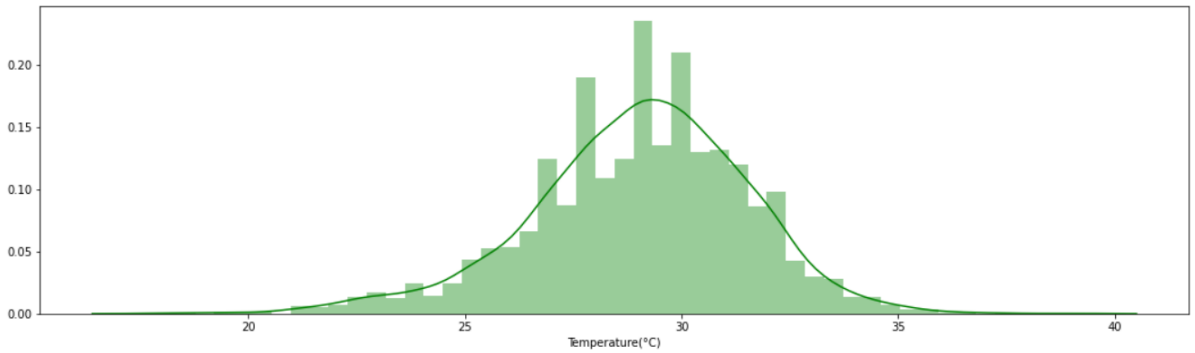
เนื่องจากการคำนวณค่าดัชนีคุณภาพน้ำ หรือ WQI ซึ่งอ้างอิงจาก National Sanitation Foundation Water Quality Index (NSF WQI) จะต้องทำการเปลี่ยนค่าคุณสมบัติของน้ำแต่ละตัวให้อยู่ในรูปแบบของ Q-value โดยค่า Q-value นี้จะมีค่าอยู่ในช่วง 0-100 โดยเราทำการสร้างคอลัมน์ใหม่ซึ่งเก็บค่าคุณสมบัติของน้ำแต่ละตัวในรูปแบบของ Q-Value (Q-value Normalization) หลังจากนั้นจึงทำการนำค่าคุณสมบัติของน้ำแต่ละมาคูณกับน้ำหนัก ซึ่งค่าคุณสมบัติของน้ำแต่ละตัวก็จะมีน้ำหนักในการคำนวณแตกต่างกันออกไป แล้วนำมารวมกันจึงจะได้ค่า WQI แล้วนำค่าดังกล่าว เปรียบเทียบตามเกณฑ์คุณภาพน้ำ เพื่อบ่งชี้ว่าคุณภาพน้ำอยู่ในเกณฑ์ ดีมาก ดี พอใช้ เสื่อมโทรม และเสื่อมโทรมมาก แต่เนื่องจากข้อมูลที่ได้มาจากกรมควบคุมมลพิษ มีเพียงค่า WQI ที่อยู่ในเกณฑ์ดี พอใช้ และเสื่อมโทรม เท่านั้น ทำให้ข้อมูลที่จะนำไปฝึกสอนจะมีเพียง 3 ประเภทดังกล่าว ซึ่งคอลัมน์ที่จะใช้เก็บข้อมูลมีชื่อว่า Class_num โดย หมายเลข 1 เท่ากับ ดี, หมายเลข 2 เท่ากับ พอใช้ และหมายเลข 3 เท่ากับเสื่อมโทรม

	Temp	pH	Tur	DO	Class_num
0	29.5	7.8	20.7	2.5	2
1	29.8	7.6	14.0	3.9	2
2	29.3	7.6	16.8	4.9	2
3	29.3	7.4	16.3	4.8	2
4	29.0	7.1	13.1	2.3	2
...	...	...	...	...	...

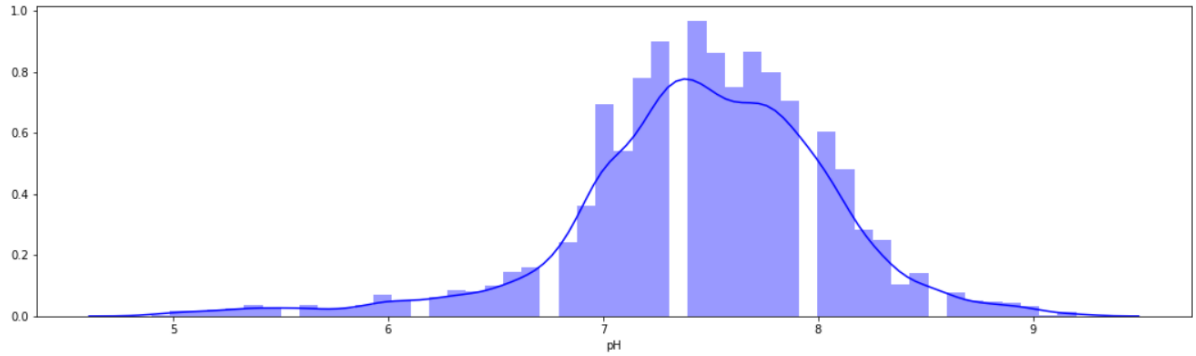
รูปที่ 9 : ข้อมูลที่ทำการคำนวณค่าดัชนีคุณภาพน้ำเพื่อจำแนกเกณฑ์ข้อมูลแล้ว

	Temp	pH	Tur	DO
count	2752.000000	2752.000000	2752.000000	2752.000000
mean	29.019586	7.458975	62.227384	5.770712
std	2.502468	0.597413	122.221913	1.767965
min	17.900000	4.900000	0.000000	0.000000
25%	27.600000	7.200000	10.200000	4.800000
50%	29.100000	7.500000	26.200000	6.000000
75%	30.700000	7.800000	60.425000	7.000000
max	39.000000	9.200000	1947.000000	13.900000

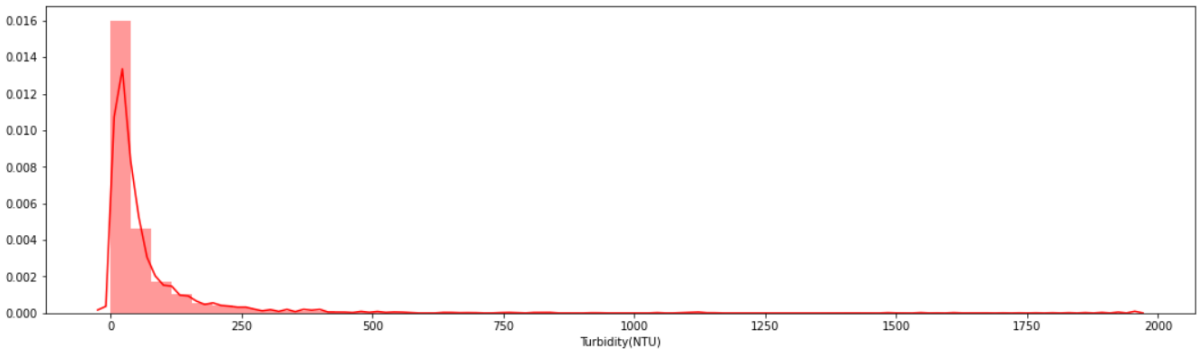
รูปที่ 10 : ตารางข้อมูลทางสถิติของข้อมูล



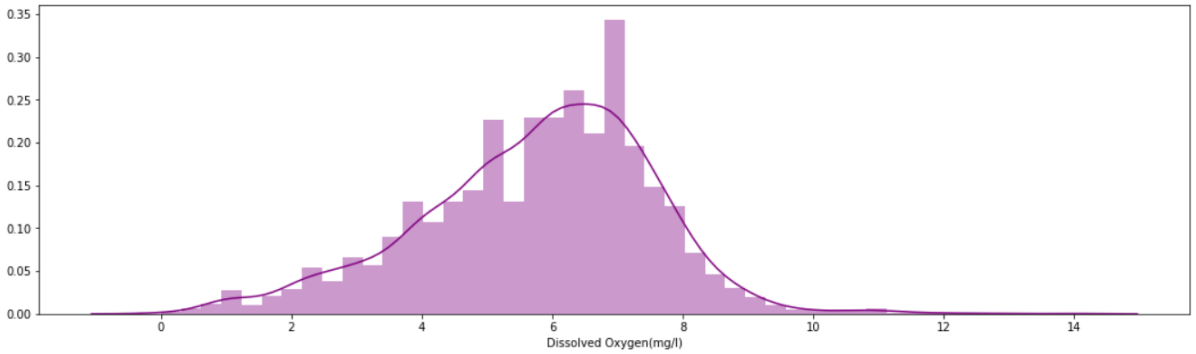
รูปที่ 11 : กราฟแสดงการกระจายของข้อมูลคุณสมบัติของน้ำ ตัวแปรอุณหภูมิ



รูปที่ 12 : กราฟแสดงการกระจายของข้อมูลคุณสมบัติของน้ำ ตัวแปรความเป็นกรด-ด่าง

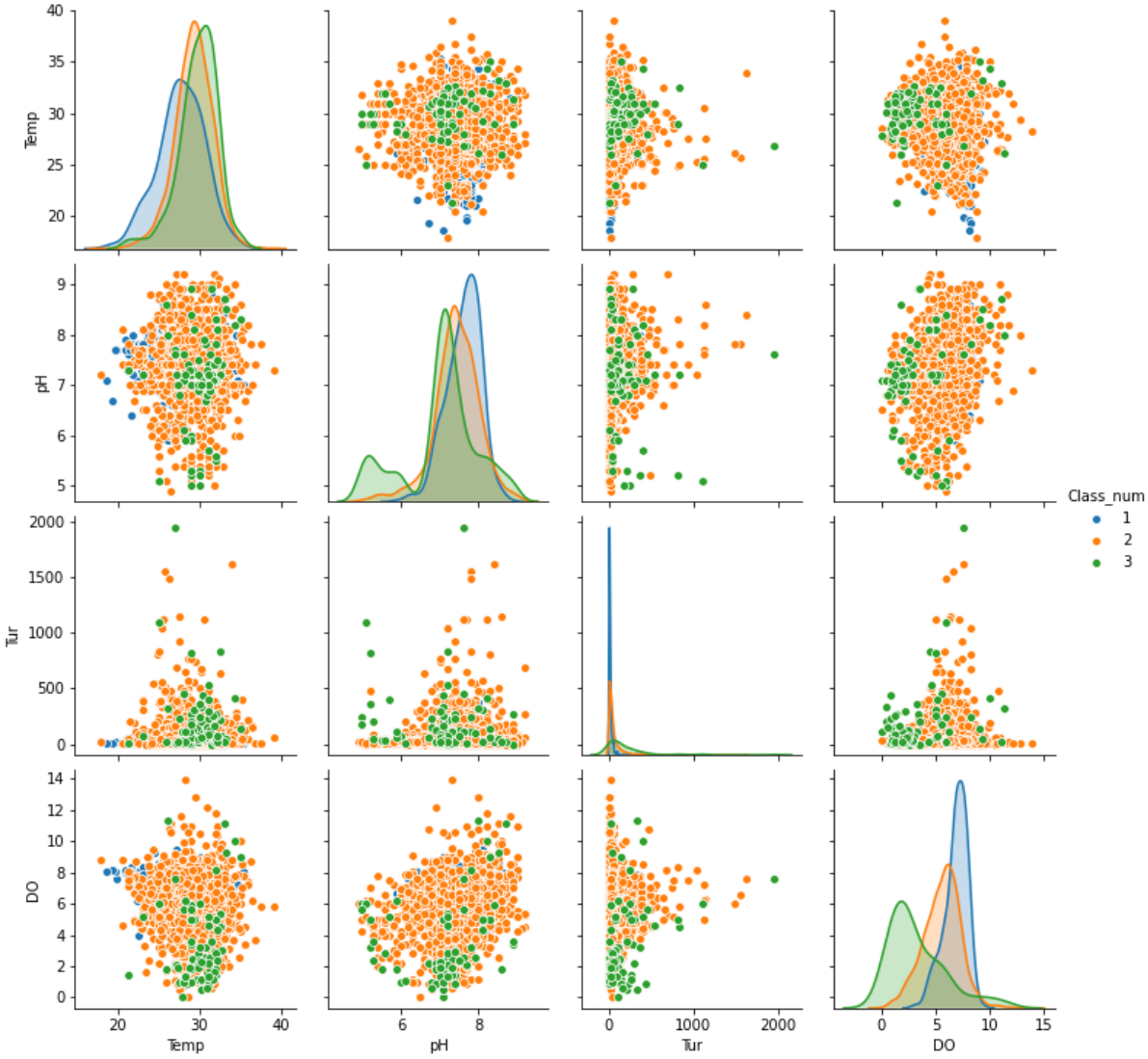


รูปที่ 13 : กราฟแสดงการกระจายของข้อมูลคุณสมบัติของน้ำ ตัวแปรความขุ่น



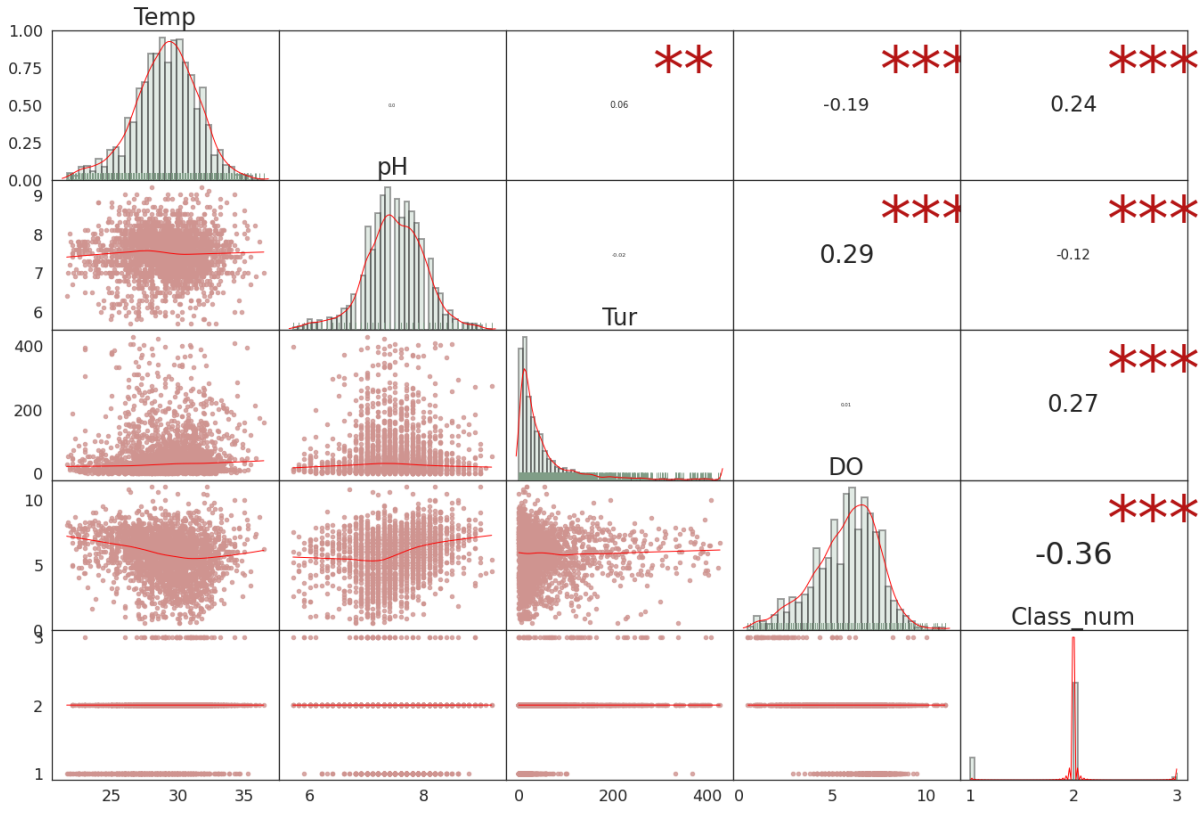
รูปที่ 14 : กราพแสดงการกระจายของข้อมูลคุณสมบัติของน้ำ ตัวแปรออกซิเจนที่ละลาย

จากรูปที่ 10 พบว่าเบื้องต้นจะสังเกตุว่าข้อมูลคุณภาพน้ำแต่ละตัวจะมีขนาดที่ค่อนข้างแตกต่างกัน โดยเฉพาะความขุ่นที่มีค่าเฉลี่ยสูงมากถึง 62 ส่วนเบี่ยงเบนมาตรฐานถึง 122.22 และนอกจากนั้นยังมีค่าต่ำสุดเป็น 0 แต่ก็มีค่าสูงสุดถึง 1947 ประกอบกับจากรูปที่ 13 จะเห็นว่าข้อมูลความขุ่นมีค่าค่อนข้างกระจายมาก ซึ่งมีความเป็นไปได้สูงที่จะมีค่าผิดปกติ (Outliers)



รูปที่ 15 : กราฟแบบจุดโดยจำแนกเกณฑ์ของข้อมูลโดยใช้สี

3.5.3 การหาความสัมพันธ์ของข้อมูล (Correlation)

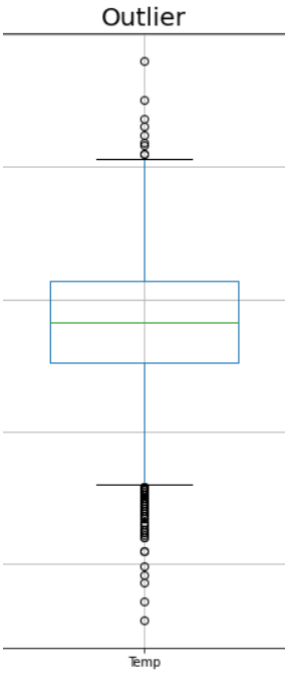


รูปที่ 16 : ความสัมพันธ์ระหว่างข้อมูลคุณสมบัติของน้ำ

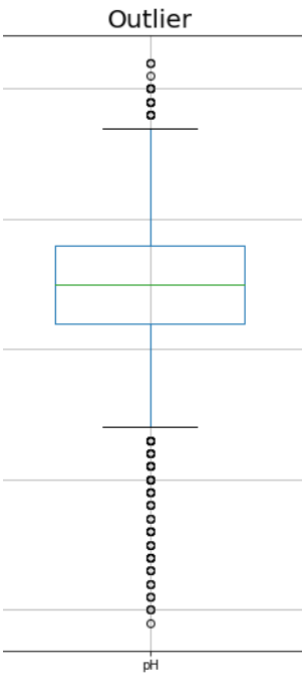
จากข้อมูลดิบที่ได้มากจากกรมควบคุมมลพิษ จะเห็นว่าข้อมูลคุณสมบัติของน้ำเป็นมีลักษณะแบ่งเป็น 2 ประเภท คือข้อมูลทางกายภาพและข้อมูลทางชีวภาพ แต่เนื่องจากวัตถุประสงค์ของโครงงาน คือต้องการที่จะนำข้อมูลทางกายภาพหรือข้อมูลที่สามารถตรวจวัดได้จากเซนเซอร์ซึ่ง ได้แก่ อุณหภูมิ, ความเป็นกรด-ด่าง, ความขุ่น และออกซิเจนที่ละลายมาใช้ในการฝึกสอน ทำให้การคัดเลือกข้อมูลคุณสมบัติของน้ำเพื่อที่จะลดขนาดของข้อมูลที่จะใช้ฝึกสอน จะพิจารณาจากข้อมูลทั้ง 4 ตัวข้างต้น โดยเมื่อทำการนำข้อมูลคุณสมบัติของน้ำมาหาความสัมพันธ์ในรูปของสัมประสิทธิ์ความสัมพันธ์ (correlation coefficient) ดังแสดงในรูปที่ 16 แสดงให้เห็นว่าข้อมูลคุณสมบัติของน้ำแต่ละตัวค่อนข้างไม่มีความสัมพันธ์กัน ประกอบกับขนาดของข้อมูลมีความไม่ใหญ่มากทำให้ผลการพิจารณาคัดเลือกข้อมูลฝึกสอน จะใช้ทั้ง 4 ตัวคือ อุณหภูมิ, ความเป็นกรด-ด่าง , ความขุ่น และออกซิเจนที่ละลาย

3.5.4 การกำจัดค่าผิดปกติ (Outlier Detection)

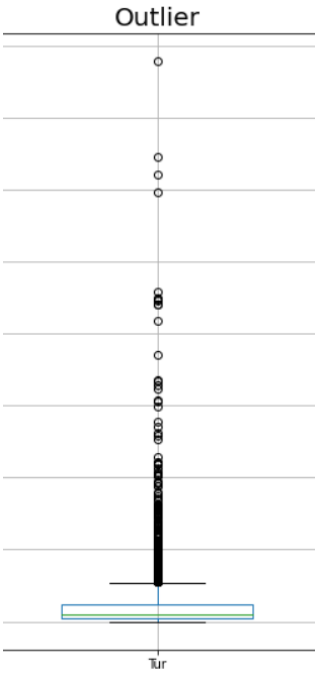
หลังจากที่ได้ประเภทของข้อมูลคุณภาพในแต่ละแถวแล้ว จากนั้นจะต้องทำการตรวจหาค่าผิดปกติของข้อมูล โดยวิธีที่จะใช้คือการแสดงข้อมูลในรูปแบบของ Boxplot



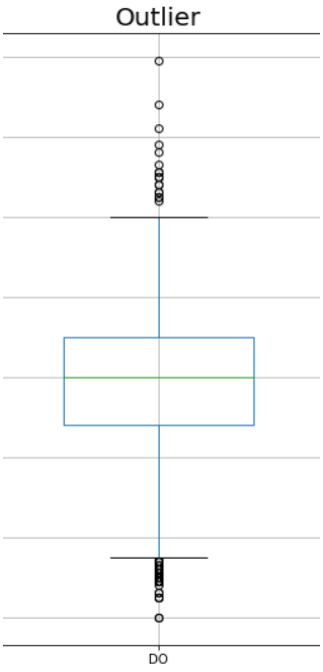
รูปที่ 17



รูปที่ 18



รูปที่ 19



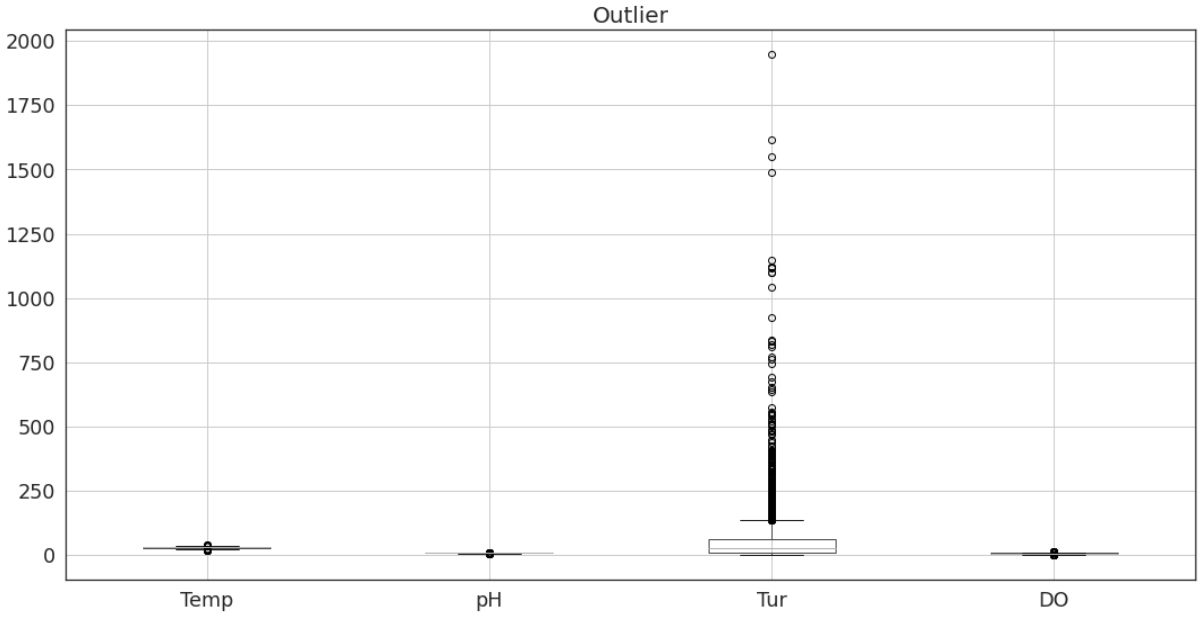
รูปที่ 20

รูปที่ 17 : การแสดงผลข้อมูลแบบกล่องของข้อมูลคุณภาพน้ำ ตัวแปรอุณหภูมิ

รูปที่ 18 : การแสดงผลข้อมูลแบบกล่องของข้อมูลคุณภาพน้ำ ตัวแปรความเป็นกรด-ด่าง

รูปที่ 19 : การแสดงผลข้อมูลแบบกล่องของข้อมูลคุณภาพน้ำ ตัวแปรความขุ่น

รูปที่ 20 : การแสดงผลข้อมูลแบบกล่องของข้อมูลคุณภาพน้ำ ตัวแปรออกซิเจนที่ละลาย



รูปที่ 21 : การแสดงผลข้อมูลแบบกล่องของข้อมูลคุณภาพน้ำโดยรวม

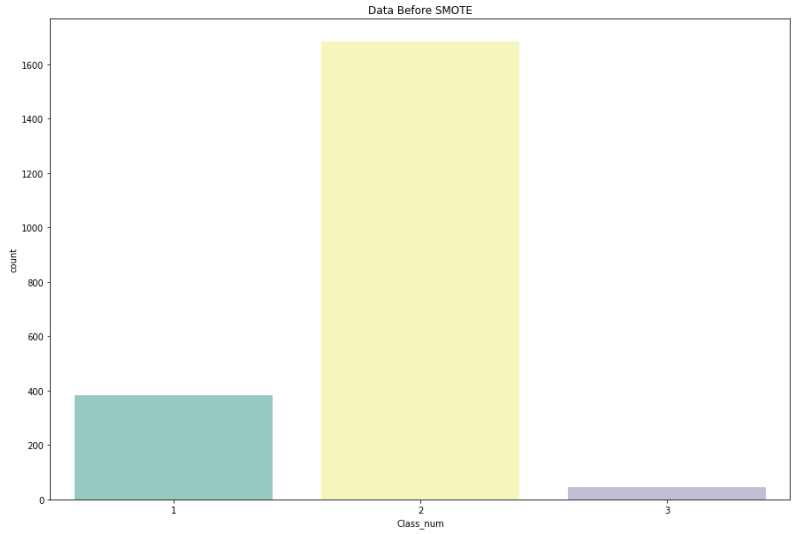
จะเห็นว่าจากรูปที่ 17-20 ข้อมูลคุณสมบัติของน้ำทุกตัวล้วนมีค่าผิดปกติ แล้วจากรูปที่ 21 ก็แสดงให้เห็นได้ชัดเจนว่าข้อมูลมีขนาดค่อนข้างต่างกันมาก ๆ จากนั้นทำการกำจัดค่าผิดปกติออกโดยอ้างอิงเกณฑ์จาก ค่า Z-score ของข้อมูลดิบ โดยที่การคำนวณ Z-score ก็คือการปรับขนาด และการหาจุดศูนย์กลางของข้อมูล ดังนั้นจุดที่อยู่ไกลจุดศูนย์กลางมาเกินไปจากถือเป็นค่าผิดปกติ ซึ่งเกณฑ์ส่วนใหญ่ จะอยู่ในช่วง [-3,3] ดังนั้นหาก Z-score มีค่ามากกว่าหรือน้อยกว่า 3 หรือ -3 ตามลำดับจะถูกคิดว่าเป็นค่าผิดปกติและถูกกำจัดออก โดยมีข้อมูลที่ผิดปกติและถูกกำตัดออก 114 แถว จึงมีข้อมูลเหลืออยู่ 2638 แถว

3.5.5 การแบ่งชุดข้อมูล

ทำการแบ่งข้อมูลเป็นชุดฝึกสอน (Train-set) และชุดทดสอบ (Test-set) โดยจะแบ่งเป็นชุดฝึกสอนร้อยละ 80 ของข้อมูล เท่ากับ 2110 แถว และชุดทดสอบเป็นร้อยละ 20 ของข้อมูล เท่ากับ 528 แถว

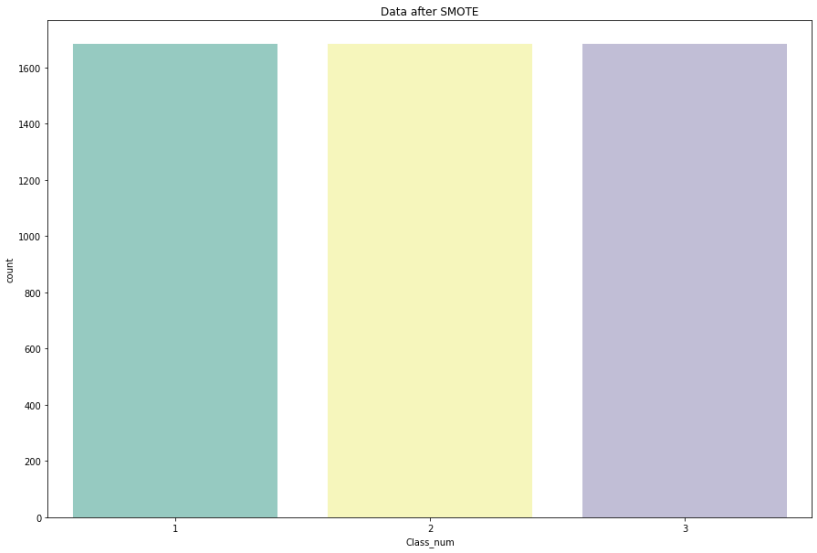
3.5.6 การปรับความสมดุลของข้อมูล

เนื่องจากข้อมูลในชุดฝึกสอนที่ผ่านการทำความสะอาดและผ่านการคัดเลือก มีความไม่สมดุล (Imbalance) ค่อนข้างมาก กล่าวคือจำนวนข้อมูลที่แบ่งตามเกณฑ์คุณภาพของน้ำแต่ละเกณฑ์ มีจำนวนค่อนข้างต่างกันมาก ดังที่เห็นได้จากรูป 22



รูปที่ 22 : จำนวนของข้อมูลจำแนกประเภทตามเกณฑ์คุณภาพน้ำก่อนสุ่มเพิ่มกลุ่มตัวอย่าง SMOTE

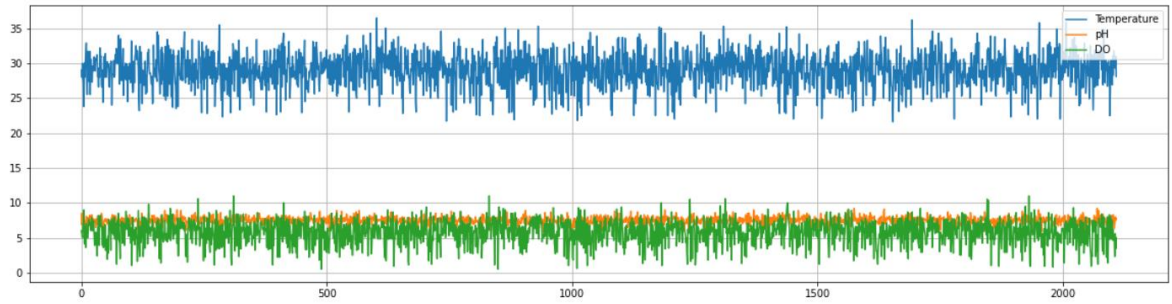
โดยข้อมูลที่เป็นเกณฑ์ดี มี 382 แถว เกณฑ์พอใช้มี 1684 แถว และเกณฑ์เสื่อมโทรมมีเพียง 44 แถว ทำให้ต้องมีการเพิ่มตัวอย่างข้อมูล ให้ในแต่ละเกณฑ์มีจำนวนข้อมูลที่เท่าหรือใกล้เคียงกันให้มากที่สุด โดยจะทำการ Over-sampling (SMOTE) คือมีการสุ่มตัวอย่างให้ข้อมูลในกลุ่มน้อย (Minor class) ในที่นี้คือเกณฑ์พอใช้ เพิ่มขึ้นให้เท่ากับข้อมูลกลุ่มหลัก (Major class) โดยหลักจากการสุ่มกลุ่มตัวอย่างทำให้มีจำนวนข้อมูลทั้งสิ้น 5052 แถว แบ่งเป็นเกณฑ์ละเท่า ๆกัน เกณฑ์ละ 1684 แถว โดยที่แสดงดังรูป 23



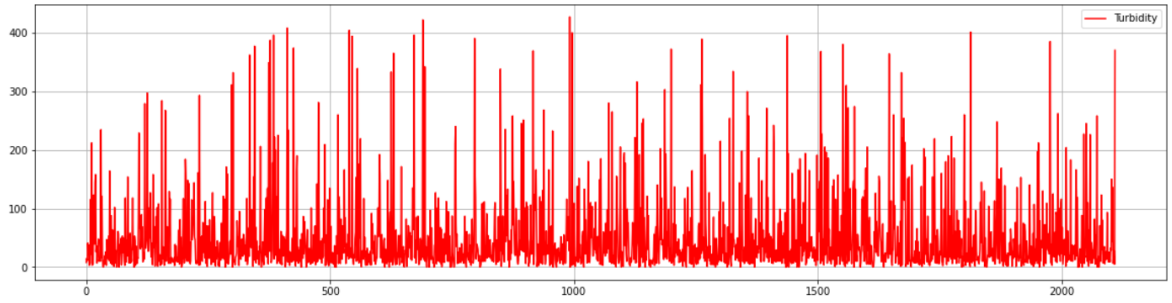
รูปที่ 23 : จำนวนของข้อมูลจำแนกประเภทตามเกณฑ์คุณภาพน้ำหลังสุ่มเพิ่มกลุ่มตัวอย่าง SMOTE

3.5.7 การปรับขนาดของข้อมูล (Normalization)

เนื่องจากข้อมูลคุณสมบัติของน้ำแต่ละตัวมีขนาดค่อนข้างแตกต่างกันมาก จากรูปที่ 24 และ 25 จะเห็นว่าช่วงของข้อมูลบางตัว เช่น ความเป็นกรด-ด่าง และออกซิเจนที่ละลาย มีค่าระหว่าง 0 ถึง 15 และอุณหภูมิก็มีค่าไม่เกิน 40 องศา แต่ในขณะเดียวกันที่ความขุ่นมีค่าค่อนข้างต่างกันมาก โดยมีค่าทีสูงที่สุดถึง 427 ดังนั้นจึงต้องมีการปรับขนาดของข้อมูลเพื่อลด bias ของข้อมูลโดยวิธีการ Minmax normalization เพื่อปรับขนาดของข้อมูลให้อยู่ในช่วง [0,1] โดยข้อมูลที่น้อยที่สุดจะเท่ากับ 0 และข้อมูลที่มากที่สุดจะเท่ากับ 1



รูปที่ 24 : ช่วงของข้อมูลคุณสมบัติของน้ำตัวแปรอุณหภูมิ ความเป็นกรด-ด่าง และออกซิเจนที่ละลาย



รูปที่ 25 : ช่วงของข้อมูลคุณสมบัติของน้ำตัวแปรความขุ่น

3.5.8 คัดเลือกพารามิเตอร์ของแบบจำลอง

Hyperparameter	Value	Best Parameter
metric	["euclidean", manhattan","minkowski"]	euclidean
n_neighbors	(range(1,100))	21
p	1,2	1
weights	"uniform", "distance"	uniform

ตารางที่ 4 : Hyperparameter ของ K-NN ที่ทำการ scaled

Hyperparameter	Value	Best Parameter
metric	["euclidean", manhattan","minkowski"]	manhattan
n_neighbors	(range(1,100))	1
p	1,2	1
weights	"uniform", "distance"	uniform

ตารางที่ 5 : Hyperparameter ของ K-NN with balancing

Hyperparameter	Value	Best Parameter
Bootstrap	true	
max_depth	2,5,10,20,50,90,100,None	5
max-feature	2,3	3
min_sample_leaf	3,4,5	3
min_sample_split	8,10,12	8
n_estimator	5,10,30,40,70,100	70

ตารางที่ 6 : Hyperparameter ของ Random Forest ที่ทำ Minmax normalization

Hyperparameter	Value	Best Parameter
Bootstrap	true	
max_depth	2,5,10,20,50,90,100,None	100
max-feature	2,3	3
min_sample_leaf	3,4,5	3
min_sample_split	8,10,12	8
n_estimator	5,10,30,40,70,100	100

ตารางที่ 7 : Hyperparameter ของ Random Forest ที่ทำ Minmax normalization และ balancing

Hyperparameter	Value	Best Parameter
C	np.logspace(0,4,num=10)	7.742636826811269
multi_class	ovr	‘ovr’
penalty	l1,l2	l1
solver	'liblinear', 'saga'	‘liblinear’

ตารางที่ 8 : Hyperparameter ของ Logistic Regression ที่ทำ Minmax normalization แต่ไม่มีการสุ่มเพิ่มกลุ่มตัวอย่าง (SMOTE)

Hyperparameter	Value	Best Parameter
C	np.logspace(0,4,num=10)	2.7825594022971245
multi_class	ovr	‘ovr’
penalty	l1,l2	l1
solver	'liblinear', 'saga'	“liblinear”

ตารางที่ 9 : Hyperparameter ของ Logistic Regression ที่ทำ Minmax normalization และมีการสุ่มเพิ่มกลุ่มตัวอย่าง (SMOTE)

Hyperparameter	Value	Best Parameter
Input layer	16,32,64,256	32
Hidden layer	16,32,64,256	64
Output layer	3	3
Epochs	30,40,50	30
batches_size	5,10,100	10
Learning rate	0.01,0.1,1	0.0.1
beta_1		0.3
beta_2		0.5

ตารางที่ 10 : Hyperparameter ของ Neural Network ที่มีการ scaler แต่ไม่มีการสุ่มเพิ่มกลุ่มตัวอย่าง (SMOTE)

Hyperparameter	Value	Best Parameter
Input layer	16,32,64,256	32
Hidden layer	16,32,64,256	256
Output layer		3
Epochs	30,40,50	50
Batches	5,10,100	100
Learning rate	0.01,0.1,1	0.1
beta_1		0.3
beta_2		0.5

ตารางที่ 11 : Hyperparameter ของ Neural Network ที่มีการ scaler และมีการสุ่มเพิ่มกลุ่มตัวอย่าง (SMOTE)

บทที่ 4 ผลการดำเนินงาน

ในโครงงานวิจัยนี้ ได้มีการนำข้อมูลคุณสมบัติของน้ำ มาทำการวิเคราะห์และทำนายเกณฑ์ของน้ำว่าอยู่ในเกณฑ์ใด โดยทางผู้จัดทำได้ทำการวิเคราะห์ข้อมูลดังกล่าวกับแบบจำลองการเรียนรู้ของเครื่องจักร จำนวน 5 แบบจำลองได้แก่ Gaussian Naive Bayes, Random Forest, K-Nearest Neighbor, Logistic Regression และ Neural Network : Multilayer Perceptron (MLP) และนำผลลัพธ์มาเปรียบเทียบประสิทธิภาพของแต่ละแบบจำลองโดยใช้ตัวชี้วัดคือ F1-Score, Precision, Recall และ Accuracy

4.1 ผลการดำเนินงาน

	precision	recall	f1-score	support
ดี	0.53383	0.72449	0.61472	98
พอใช้	0.90217	0.79616	0.84586	417
เสื่อมโทรม	0.14815	0.30769	0.20000	13
accuracy			0.77083	528
macro avg	0.52805	0.60945	0.55353	528
weighted avg	0.81524	0.77083	0.78706	528

ตารางที่ 12 : รายงานการแบ่งกลุ่มของแบบจำลอง Random Forest (With SMOTE)

	precision	recall	f1-score	support
ดี	0.66667	0.36735	0.47368	98
พอใช้	0.84534	0.95683	0.89764	417
เสื่อมโทรม	1.00000	0.15385	0.26667	13
accuracy			0.82765	528
macro avg	0.83734	0.49268	0.54600	528
weighted avg	0.81598	0.82765	0.80341	528

ตารางที่ 13 : รายงานการแบ่งกลุ่มของแบบจำลอง Random Forest (Without SMOTE)

	precision	recall	f1-score	support
ดี	0.49505	0.51020	0.50251	98
พอใช้	0.85930	0.82014	0.83926	417
เสื่อมโทรม	0.17241	0.38462	0.23810	13
accuracy			0.75189	528
macro avg	0.50892	0.57165	0.52662	528
weighted avg	0.77478	0.75189	0.76196	528

ตารางที่ 14 : รายงานการแบ่งกลุ่มของแบบจำลอง K-Nearest Neighbor (With SMOTE)

	precision	recall	f1-score	support
ดี	0.64516	0.40816	0.50000	98
พอใช้	0.84946	0.94724	0.89569	417
เสื่อมโทรม	1.00000	0.07692	0.14286	13
accuracy			0.82576	528
macro avg	0.83154	0.47744	0.51285	528
weighted avg	0.81525	0.82576	0.80371	528

ตารางที่ 15 : รายงานการแบ่งกลุ่มของแบบจำลอง K-Nearest Neighbor (Without SMOTE)

	precision	recall	f1-score	support
ดี	0.42714	0.86735	0.57239	98
พอใช้	0.93981	0.48681	0.64139	417
เสื่อมโทรม	0.08850	0.76923	0.15873	13
accuracy			0.56439	528
macro avg	0.48515	0.70780	0.45750	528
weighted avg	0.82370	0.56439	0.61670	528

ตารางที่ 16 : รายงานการแบ่งกลุ่มของแบบจำลอง Logistic Regression (With SMOTE)

	precision	recall	f1-score	support
ดี	0.60000	0.18367	0.28125	98
พอใช้	0.81325	0.97122	0.88525	417
เสื่อมโทรม	0.00000	0.00000	0.00000	13
accuracy			0.80114	528
macro avg	0.47108	0.38497	0.38883	528
weighted avg	0.75365	0.80114	0.75134	528

ตารางที่ 17 : รายงานการแบ่งกลุ่มของแบบจำลอง Logistic Regression (Without SMOTE)

	precision	recall	f1-score	support
ดี	0.35983	0.87755	0.51039	98
พอใช้	0.93158	0.42446	0.58320	417
เสื่อมโทรม	0.10101	0.76923	0.17857	13
accuracy			0.51705	528
macro avg	0.46414	0.69041	0.42405	528
weighted avg	0.80501	0.51705	0.55972	528

ตารางที่ 18 : รายงานการแบ่งกลุ่มของแบบจำลอง Gaussian Naive Bayes (With SMOTE)

	precision	recall	f1-score	support
ดี	0.57273	0.64286	0.60577	98
พอใช้	0.88725	0.86811	0.87758	417
เสื่อมโทรม	0.20000	0.15385	0.17391	13
accuracy			0.80871	528
macro avg	0.55333	0.55494	0.55242	528
weighted avg	0.81196	0.80871	0.80980	528

ตารางที่ 19 : รายงานการแบ่งกลุ่มของแบบจำลอง Gaussian Naive Bayes (Without SMOTE)

	precision	recall	f1-score	support
ดี	0.60000	0.70000	0.65000	98
พอใช้	0.91000	0.83000	0.87000	417
เสื่อมโทรม	0.22000	0.54000	0.31000	13
accuracy			0.80000	528
macro avg	0.58000	0.69000	0.61000	528
weighted avg	0.83000	0.80000	0.81000	528

ตารางที่ 20 : รายงานการแบ่งกลุ่มของแบบจำลอง Neural Network :MLP (With SMOTE)

	precision	recall	f1-score	support
ดี	0.68000	0.52000	0.59000	98
พอใช้	0.87000	0.94000	0.90000	417
เสื่อมโทรม	0.00000	0.00000	0.00000	13
accuracy			0.84000	528
macro avg	0.52000	0.49000	0.50000	528
weighted avg	0.81000	0.84000	0.82000	528

ตารางที่ 21 : รายงานการแบ่งกลุ่มของแบบจำลอง Neural Network :MLP (Without SMOTE)

	Gaussian Naive Bayes	Random Forest	K-Nearest Neighbor	Logistic Regression	Neural Network : MLP
Precision	0.55333	0.83734	0.83154	0.47108	0.5200
Recall	0.55494	0.49268	0.47744	0.38497	0.4900
F1-Score	0.55242	0.54600	0.51285	0.38883	0.5000
Accuracy	0.80871	0.82765	0.82576	0.80114	0.8400

ตารางที่ 22 : การเปรียบเทียบตัวชี้วัดประสิทธิภาพของแบบจำลองการทำนายโดยไม่มีการสุ่มเพิ่มกลุ่มตัวอย่าง (SMOTE)

จากตารางที่ 22 จะเห็นได้ว่าแบบจำลอง Neural Network : Multilayer Perceptron (MLP) มีประสิทธิภาพในการทำนายเกณฑ์คุณภาพน้ำโดยรวมมากที่สุดเมื่อเทียบกับแบบจำลองอื่นๆ โดยได้ค่าความถูกต้อง (Accuracy) ที่มากที่สุดคือ 0.84 ส่วนแบบจำลอง Random Forest ได้ค่าความแม่นยำ (Precision) มากที่สุดคือ 0.83734 แบบจำลอง K-Nearest Neighbor ได้ค่า Recall และ F1-Score ที่มากที่สุด คือ 0.55494 และ 0.55242 ตามลำดับ

	Gaussian Naive Bayes	Random Forest	K-Nearest Neighbor	Logistic Regression	Neural Network : MLP
Precision	0.46414	0.52805	0.50892	0.48515	0.5800
Recall	0.69041	0.60945	0.57165	0.70780	0.6900
F1-Score	0.42405	0.55353	0.52662	0.45750	0.6100
Accuracy	0.51705	0.77083	0.75189	0.56439	0.8000

ตารางที่ 23 : การเปรียบเทียบตัวชี้วัดประสิทธิภาพของแบบจำลองการทำนายโดยมีการสุ่มเพิ่มกลุ่มตัวอย่าง (SMOTE)

จากตารางที่ 23 จะเห็นได้ว่าหลังจากทำการสุ่มเพิ่มกลุ่มตัวอย่าง (SMOTE) พบว่าแบบจำลอง Neural Network : Multilayer Perceptron (MLP) ยังคงมีประสิทธิภาพในการทำนายเกณฑ์คุณภาพน้ำโดยรวมมากที่สุดเมื่อเทียบกับแบบจำลองอื่นๆ โดยได้ค่าความถูกต้อง ความแม่นยำ และ F1-Score ที่มากที่สุดคือ 0.80 , 0.58 และ 0.61 ตามลำดับ ส่วนแบบจำลอง Logistic Regression ได้ค่า Recall ที่มากที่สุด คือ 0.70780

เนื่องจากวัตถุประสงค์ของโครงงานวิจัย ต้องการที่จะสร้างแบบจำลองการทำนายคุณภาพน้ำเพื่อที่จะสามารถหาแนวทางในการแก้ปัญหาคุณภาพน้ำเสื่อมโทรมได้อย่างรวดเร็ว ทำให้การประเมินแบบจำลองเพียงการดูค่าความถูกต้องโดยรวม (Accuracy) อาจทำให้ไม่สามารถที่จะนำไปประยุกต์ใช้กับสถานการณ์จริงได้อย่างเหมาะสมมากนัก หากแต่จะต้องพิจารณาจากค่า Precision, Recall และ F1-Score ด้วย โดยให้กลุ่มที่สนใจเป็นกลุ่มข้อมูลหมายเลย 3 หรือเกณฑ์คุณภาพน้ำเสื่อมโทรม เพื่อที่จะวัดประสิทธิภาพว่าถ้าหากเกิดการนำแบบจะลองไปประยุกต์ใช้งานจริง เช่น การนำไปใช้กับสถานีตรวจวัดคุณภาพน้ำ จะสามารถทำนายทำนายกลุ่มที่มีความสำคัญและตรงกับวัตถุประสงค์ที่สุดซึ่งในที่นี้คือ กลุ่มเกณฑ์คุณภาพน้ำเสื่อมโทรม ได้อย่างมีประสิทธิภาพ

โดยจากที่กล่าวมาข้างต้นทำให้เห็นว่า เมื่อพิจารณาเปรียบเทียบประสิทธิภาพของแบบจำลองทั้งหมดทั้งกับข้อมูลที่ไม่ได้ทำการเพิ่มกลุ่มตัวอย่าง (SMOTE) และกลุ่มที่ทำการเพิ่มกลุ่มตัวอย่างแล้ว ทำให้สรุปได้ว่าแบบจำลองการทำนาย Neural Network : Multilayer Perceptron (MLP) มีประสิทธิภาพโดยรวมดีที่สุด โดยเฉพาะค่าจากตัวชี้วัดคุณภาพน้ำในกลุ่มเกณฑ์เสื่อมโทรม โดยมีค่า Precision, Recall, F1-Score และ Accuracy เท่ากับ 0.58, 0.69, 0.61 และ 0.80 ตามลำดับ

บทที่ 5 สรุปผล อภิปรายผล และข้อเสนอแนะ

5.1 สรุปผลการดำเนินงาน

โครงงานวิจัยนี้ได้นำเสนอการใช้แบบจำลองการเรียนรู้ของเครื่อง (Machine Learning Models) ในการวิเคราะห์และทำนายเกณฑ์คุณภาพน้ำ โดยได้มีกระบวนการในการทำงานในการตรวจสอบและวิเคราะห์ข้อมูลรวมถึงไปถึงการปรับแต่งข้อมูลให้มีความเหมาะสมกับการทำงาน และการสุ่มกลุ่มตัวอย่างเพิ่มเพื่อที่จะให้ข้อมูลในแต่ละกลุ่มมีความสมดุลกัน เพื่อประสิทธิภาพที่ดีของแบจำลองการเรียนรู้ โดยจากการเปรียบประสิทธิภาพของแบบจำลองที่นำเสนอทั้ง 5 แบบจำลอง คือ Gaussian Naive Bayes ,Random Forest, K-Nearest Neighbor, Logistic Classification และ Neural Network : Multilayer Perceptron (MLP) พบว่าแบบจำลองที่มีประสิทธิภาพในการทำนายมากที่สุดคือ Neural Network : Multilayer Perceptron (MLP) โดยมีค่า Macro average เป็น Precision เท่ากับ 0.58, Recall เท่ากับ 0.69, F1-Score เท่ากับ 0.61 และ Accuracy เท่ากับ 0.80

5.2 ปัญหาและอุปสรรคในการดำเนินโครงงาน

5.2.1 ชุดข้อมูลที่ได้มามีข้อมูลบางส่วนขาดหายไปค่อนข้างมาก

5.2.2 ข้อมูลที่ใช้มีปัญหาเรื่องข้อมูลไม่สมดุลกัน (Imbalanced Dataset) ทำให้ต้องแก้ไขโดยการสุ่มกลุ่มตัวอย่างเพิ่ม

5.2.3 ใช้เวลาในการ Tune Hyperparameter ค่อนข้างนาน

5.2.4 เนื่องจากชุดข้อมูลมีจำนวนข้อมูลน้อยจึงทำให้ข้อมูลเกณฑ์คุณภาพน้ำมีไม่ครบทุกเกณฑ์ โดยมีเพียง 3 จาก 5 เกณฑ์

5.3 ข้อเสนอแนะ

5.3.1 เนื่องจากผลลัพธ์ที่ได้นั้นไม่ดีเท่าที่ควร จำเป็นต้องหาแบบจำลองอื่น ๆ เพิ่มในการวิเคราะห์และทำนาย

5.3.2 เนื่องจากข้อมูลคุณภาพน้ำนั้นมีปริมาณที่ไม่เพียงพอต่อการนำไปวิเคราะห์ ควรหาข้อมูลคุณภาพน้ำเพิ่มเติมเพื่อเพิ่มประสิทธิภาพของแบบจำลอง

5.3.3 สามารถนำไปประยุกต์ใช้กับสถานะจริงได้ เช่น นำไปประยุกต์กับเทคโนโลยี IoT โดยอาจจะรับข้อมูลคุณสมบัติของน้ำแบบเรียลไทม์ จากนั้นทำการวิเคราะห์และทำนายผลเกณฑ์คุณภาพน้ำทันที โดยจะทำให้สามารถแก้ไขปัญหาคุณภาพน้ำได้อย่างรวดเร็ว

บรรณานุกรม

[1] B. BANGKOK, “กทม. เล็งเก็บค่าบำบัดน้ำเสียภายในปี 62 เช็กบิล-บ้าน-คอนโดฯ-อพาร์ตเมนต์ 21 เขตทั่วกรุง,” 18 เมษายน 2562. [ออนไลน์]. Available: https://today.line.me/th/pc/article/กทม+เล็งเก็บค่าบำบัดน้ำเสีย+ภายในปี+62+เช็กบิล+บ้าน+คอนโดฯ+อพาร์ตเมนต์+21+เขตทั่วกรุง-2E2vp8. [%1 ที่เข้าถึง14 เมษายน 2563].

[2] Di Wu,Hadi Mohammed,Hao Wang,Razak Seidu, “Smart Data Analysis for Water Quality in Catchment Area Monitoring,” IEEE, 2019.

[3] Mohammad Salah Uddin Chowdury, Talha Bin Emran,Subhasish Ghosh,Abhijit Pathak,Mohd. Manjur Alam,Nurul Absar,Mohammad Shahadat Hossain, “IoT Based Real-time River Water Quality Monitoring System,” ScienceDirect, 2019.

[4] Ping Liu, Jin Wang,Arun Kumar Sangaiah,Yang Xie,Xinchun Yin, “Analysis and Prediction of Water Quality Using LSTM Deep Neural Networks in IoT Environment,” MDPI, 2019.

[5] Kofi Sarpong Adu-Manu, Cristiano Tapparello, Wendi Heinzelman, , “Water Quality Monitoring Using Wireless Sensor Networks: Current Trends and Future Research Directions,” ACM Transactions on Sensor Networks, 2017.

[6] Umair Ahmed, Rafia Mumtaz, Hirra Anwar, Asad A. Shah, Rabia Irfan, José García-Nieto, “Efficient Water Quality Prediction Using Supervised Machine Learning,” MDPI, 2019.

[7] A.K. Thukral, Renu Bhardwaj, Rupinder Kaur, “Water Quality Indices,” Ministry of Statistics and Programme Implementation, 2005.

[8] V. Minaphinant, “Machine Learning คืออะไร?,” 28 Febuary 2018. [ออนไลน์]. Available: https://blog.finnomena.com/machine-learning-คืออะไร-fa8bf6663c07. [%1 ที่เข้าถึง20 April 2020].

[9] M. T. Jones, “Data and analytics,” 01 Febuary 2018. [ออนไลน์]. Available: https://www.ibm.com/developerworks/library/ba-intro-data-science-1/index.html. [%1 ที่เข้าถึง15 April 2020].

[10] “การเรียนรู้แบบเบย์ (Bayesian Learning),” [ออนไลน์]. Available: https://mis.csit.sci.tsu.ac.th/noppamas/download/DataMining/DataMiningCh7V1.pdf. [%1 ที่เข้าถึง18 April 2020].

[11] N. Chakrabarty, “Implementation of Gaussian Naive Bayes in Python from scratch,” 23 January 2019. [ออนไลน์]. Available: https://hackernoon.com/implementation-of-gaussian-naive-bayes-in-python-from-scratch-c4ea64e3944d. [%1 ที่เข้าถึง14 April 2020].

[12] W. Daroontham, “รู้จัก Decision Tree, Random Forest, และ XGBoost!!! — PART 1,” 11 November 2018. [ออนไลน์]. Available: https://medium.com/@witchapongdaroontham/รู้จัก-decision-tree-random-forrest-และ-xgboost-part-1-cb49c4ac1315. [%1 ที่เข้าถึง18 April 2020].

[13] วิกิพีเดีย, “โครงข่ายประสาทเทียม,” [ออนไลน์]. Available: https://th.wikipedia.org/wiki/โครงข่ายประสาทเทียม. [%1 ที่เข้าถึง18 April 2020].

[14] “การจำแนกประเภทและการทำนายข้อมูล (Classification and Prediction),” [ออนไลน์]. Available: https://staff.informatics.buu.ac.th/~komate/886464/%5B6%5D-Classification.pdf. [%1 ที่เข้าถึง18 เมษายน 2563].

[15] “7.1 Logistic Regression Analysis คืออะไร,” [ออนไลน์]. Available: https://sites.google.com/site/mystatistics01/chapter7/logistic-regression. [%1 ที่เข้าถึง14 เมษายน 2563].

[16] ภ. ปาลวิสุทธิ์, “การเพิ่มประสิทธิภาพเทคนิคต้นไม้ตัดสินใจบนชุดข้อมูลที่ไม่สมดุลโดยวิธีการสุ่มเพิ่มตัวอย่างกลุ่มน้อยสำหรับข้อมูลการเป็นโรคติดอินเทอร์เน็ต (Improving Decision Tree Technique in Imbalanced Data Sets Using SMOTE for Internet Addiction Disorder Data),” 2016.

[17] chengz, “วัดประสิทธิภาพ Model จาก Confusion Matrix,” 03 October 2019. [ออนไลน์]. Available: https://medium.com/@cheng3374/วัดประสิทธิภาพ-model-จาก-confusion-matrix-69d391bcd48. [%1 ที่เข้าถึง16 April 2020].

ภาคผนวก

ชื่อไฟล์ : Project

ภาษาที่ใช้ : Python

คำอธิบาย : นำข้อมูลที่วิเคราะห์แล้วเข้าสู่แบบจำลองต่างๆเพื่อเปรียบเทียบประสิทธิภาพของแต่ละแบบจำลอง

Random Forest GridSearchCV

```
rfc = RandomForestClassifier()
parameters = {
    'bootstrap': [True],
    "n_estimators":[5,10,30,40,70,100],
    "max_depth":[2,5,10,20,50,90,100,None],
    'max_features': [2, 3],
    'min_samples_leaf': [3, 4, 5],
    'min_samples_split': [8, 10, 12]
}
from sklearn.model_selection import GridSearchCV
cv = GridSearchCV(rfc,parameters,cv=4)
cv.fit(X_train_scaled,y_train)

def display(results):
    print(f'Best parameters are: {results.best_params_}')
    print("\n")
    mean_score = results.cv_results_['mean_test_score']
    std_score = results.cv_results_['std_test_score']
    params = results.cv_results_['params']
    for mean,std,params in zip(mean_score,std_score,params):
        print(f'{round(mean,3)} + or -{round(std,3)} for the {params}')
display(cv)
```

Random Forest Model without SMOTE

```
from sklearn.datasets import make_classification
from sklearn.ensemble import RandomForestClassifier

# Create the model
model = RandomForestClassifier(bootstrap = True,max_depth = 5,max_features = 2,
min_samples_leaf = 3, min_samples_split = 12, n_estimators =  40)

# Fit on training data
model.fit(X_train_scaled, y_train)

from sklearn.metrics import classification_report
sk_report = classification_report(
    digits=5,
    y_true=y_test,
    y_pred=model.predict(X_test_scaled))
print(sk_report)
```

Random Forest Model with SMOTE

```
from sklearn.datasets import make_classification
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(bootstrap = True,max_depth = 100,max_features = 3,
min_samples_leaf = 3, min_samples_split = 8, n_estimators =  100)
# Fit on training data
model.fit(X_train_balanced, y_train_balanced)

from sklearn.metrics import classification_report
sk_report = classification_report(
    digits=5,
    y_true=y_test,
    y_pred=model.predict(X_test_scaled))
print(sk_report)
```

Gussian Naive Bayes without SMOTE

```
#Import Gaussian Naive Bayes model
from sklearn.naive_bayes import GaussianNB
#Create a Gaussian Classifier
gnb = GaussianNB()
#Train the model using the training sets
gnb.fit(X_train_scaled, y_train)
#Predict the response for test dataset
gnb.predict(X_test_scaled)
from sklearn.metrics import classification_report
sk_report = classification_report(
    digits=5,
    y_true=y_test,
    y_pred=gnb.predict(X_test_scaled))
print(sk_report)
```

Gussian Naive Bayes with SMOTE

```
#Import Gaussian Naive Bayes model
from sklearn.naive_bayes import GaussianNB
#Create a Gaussian Classifier
gnb = GaussianNB()
#Train the model using the training sets
gnb.fit(X_train_balanced, y_train_balanced)
#Predict the response for test dataset
gnb.predict(X_test_scaled)

from sklearn.metrics import classification_report
sk_report = classification_report(
    digits=5,
    y_true=y_test,
    y_pred=gnb.predict(X_test_scaled))
print(sk_report)
```

K-nearest neighbor GridSearchCV

```
from sklearn.ensemble import RandomForestClassifier,RandomForestRegressor
rfc = KNeighborsClassifier()
k_range = list(range(1,50))
weight_options = ["uniform", "distance"]

parameters = dict(n_neighbors = k_range, weights = weight_options)

from sklearn.model_selection import GridSearchCV
cv = GridSearchCV(rfc,parameters,cv=4)
cv.fit(X_train,y_train)

def display(results):
    print(f'Best parameters are: {results.best_params_}')
    print("\n")
    mean_score = results.cv_results_['mean_test_score']
    std_score = results.cv_results_['std_test_score']
```

K-nearest neighbor with SMOTE

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=1,weights='uniform',metric='manhattan',p= 1)
knn.fit(X_train_balanced, y_train_balanced)
from sklearn.metrics import classification_report
sk_report = classification_report(
    digits=5,
    y_true=y_test,
    y_pred=knn.predict(X_test_scaled))
print(sk_report)
    params = results.cv_results_['params']
    for mean,std,params in zip(mean_score,std_score,params):
        print(f'{round(mean,3)} + or -{round(std,3)} for the {params}')
display(cv)
```

K-nearest neighbor without SMOTE

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=1,weights='uniform',metric='manhattan',p= 1)
knn.fit(X_train_scaled, y_train)
from sklearn.metrics import classification_report
sk_report = classification_report(
    digits=5,
    y_true=y_test,
    y_pred=knn.predict(X_test_scaled))
print(sk_report)

display(cv)
```

Logistic Regression GridSearchCV

```
from sklearn.ensemble import RandomForestClassifier,RandomForestRegressor
rfc = LogisticRegression()
C = np.logspace(0, 4, num=10)
penalty = ['l1', 'l2']
solver = ['liblinear', 'saga']
multi_class =['ovr']
parameters = dict(C=C, penalty=penalty, solver=solver,multi_class = multi_class)
from sklearn.model_selection import GridSearchCV
cv = GridSearchCV(rfc,parameters,cv=4)
cv.fit(X_train_scaled,y_train)

def display(results):
    print(f'Best parameters are: {results.best_params_}')
    print("\n")
    mean_score = results.cv_results_['mean_test_score']
    std_score = results.cv_results_['std_test_score']
    params = results.cv_results_['params']
    for mean,std,params in zip(mean_score,std_score,params):
        print(f'{round(mean,3)} + or -{round(std,3)} for the {params}')
display(cv)
```

Logistic Regression without SMOTE

```
# Logistic Scale without Imbalanced
# Best parameters are: {'C': 7.742636826811269, 'multi_class': 'ovr', 'penalty': 'l1', 'solver':
'liblinear'}
from sklearn.datasets import make_classification
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression

logisticRegr = LogisticRegression(C = 7.742636826811269, multi_class = 'ovr', penalty =  'l1',
solver = 'liblinear')
logisticRegr.fit(X_train_scaled, y_train)

from sklearn.metrics import classification_report
sk_report = classification_report(
    digits=5,
    y_true=y_test,
    y_pred=logisticRegr.predict(X_test_scaled))
print(sk_report)
```

Logistic Regression with SMOTE

```
# Logistic Scale with Imbalanced
from sklearn.datasets import make_classification
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
 logisticRegr = LogisticRegression(C = 2.7825594022071245, multi_class = 'ovr', penalty =  'l1',
 solver = 'liblinear')
logisticRegr.fit(X_train_balanced, y_train_balanced)

from sklearn.metrics import classification_report
sk_report = classification_report(
    digits=5,
    y_true=y_test,
    y_pred=logisticRegr.predict(X_test_scaled))
print(sk_report)
```

Neural Network GridSearchCV

```
# Set random seed
np.random.seed(0)
# Create function returning a compiled network
def create_network(learn_rate=0.01, momentum=0,neuron=64,neuron2=64):
    print("iteration")
    # Start neural network
    network = models.Sequential()
        network.add(layers.Dense(units=neuron, activation='relu', input_shape=(4,))
    network.add(layers.Dense(units=neuron2, activation='relu')
    network.add(layers.Dense(units=3, activation='softmax'))
    # Compile neural network
    network.compile(loss='categorical_crossentropy', # Cross-entropy
                    optimizer=adam(lr=learn_rate, beta_1=0.3, beta_2=0.4, amsgrad=False),
    # Optimizer
                    metrics=['accuracy']) # Accuracy performance metric
    return network
neural_network = KerasClassifier(build_fn=create_network, verbose)
learn_rate = [0.01,0.1]
# Create hyperparameter space
neuron = [32,256]
neuron2 = [64,256]
epochs = [30,40,50]
batches = [5, 10, 100]
# optimizers = ['rmsprop', 'adam']
# Create hyperparameter options
hyperparameters = dict(epochs=epochs,
batch_size=batches,learn_rate=learn_rate,neuron=neuron,neuron2=neuron2)
# Create grid search
grid = GridSearchCV(estimator=neural_network, cv=3, param_grid=hyperparameters)
# Fit grid search
grid_result = grid.fit(X_train_scaled, y_train_oh)
print(grid_result.best_params_)

)
```

Neural Network without SMOTE

```
#MinMax scale without SMOTE
# {'batch_size': 100, 'epochs': 30, 'learn_rate': 0.01, 'neuron': 32, 'neuron2': 64}
model = Sequential()
model.add(Dense(32, input_dim=4, activation='relu'))
# model.add(Dense(16, activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(3, activation='softmax'))
model.summary()

from keras.optimizers import adam
adam=adam(lr=0.01, beta_1=0.3, beta_2=0.4, amsgrad=False)
model.compile(loss='categorical_crossentropy', optimizer= adam, metrics=['accuracy'])
# monitor =EarlyStopping(monitor = 'loss',min_delta=1e-3,patience = 5,verbose =
0,restore_best_weights=True)
model.fit(X_train_scaled, y_train_oh, batch_size=10, epochs=30, verbose=1)
scores = model.evaluate(X_test_scaled, y_test_oh, verbose=0)

# evaluate the network
print("[INFO] evaluating network...")
predictions = model.predict(X_test_scaled, batch_size=16)
print(classification_report(y_test_oh.argmax(axis=1),predictions.argmax(axis=1),
target_names=["1","2","3"]))
```

Neural Network with SMOTE

```
# {'batch_size': 100, 'epochs': 40, 'learn_rate': 0.01, 'neuron': 32, 'neuron2': 256}
model = Sequential()
model.add(Dense(32, input_dim=4, activation='relu'))
# model.add(Dense(16, activation='relu'))
model.add(Dense(256, activation='relu'))
model.add(Dense(3, activation='softmax'))
model.summary()
from keras.optimizers import adam
adam=adam(lr=0.01, beta_1=0.3, beta_2=0.4, amsgrad=False)
model.compile(loss='categorical_crossentropy', optimizer= adam, metrics=['accuracy'])
# monitor =EarlyStopping(monitor = 'loss',min_delta=1e-3,patience = 5,verbose =
0,restore_best_weights=True)
model.fit(X_train_balanced, y_train_balanced_oh, batch_size=100, epochs=40, verbose=1)
scores = model.evaluate(X_test_scaled, y_test_oh, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
# evaluate the network
print("[INFO] evaluating network...")
predictions = model.predict(X_test_scaled, batch_size=16)
print(classification_report(y_test_oh.argmax(axis=1),predictions.argmax(axis=1),
target_names=["1","2","3"]))
```

ชื่อไฟล์ : EDA

ภาษา : python

คำอธิบาย : โหลดและวิเคราะห์ข้อมูลเพื่อคัดเลือกและปรับแต่งข้อมูลที่จะใช้ในแบบจำลองการเรียนรู้

```
# import Library
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy import stats
from imblearn.over_sampling import SMOTE
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import RobustScaler
from sklearn.model_selection import train_test_split
%matplotlib inline
# Load Data
df = pd.read_csv('/content/dataforcode1.csv')
df.drop(['??????','????????','Station','Year','Date','Time','Temp(a)','Cond','TCB','NH3-N','SS','TDS'], 1,
inplace=True)
df.rename(columns={'Temp(w)':'Temp'},inplace=True)
# Outlier Detection
z_scores = stats.zscore(data)
abs_z_scores = np.abs(z_scores)
filtered_entries = (abs_z_scores < 3).all(axis=1)
data_rm_outlier = data[filtered_entries]
print(data_rm_outlier.groupby('Class_num').size())
# Data Correlation
def corrdot(*args, **kwargs):
    corr_r = args[0].corr(args[1], 'pearson')
    corr_text = round(corr_r, 2)
    ax = plt.gca()
    font_size = abs(corr_r) * 80 + 5
    ax.annotate(corr_text, [.5, .5,],  xycoords="axes fraction",
                ha='center', va='center', fontsize=font_size)
```

```
def corrfunc(x, y, **kws):
    r, p = stats.pearsonr(x, y)
    p_stars = ''
    if p <= 0.05:
        p_stars = '*'
    if p <= 0.01:
        p_stars = '**'
    if p <= 0.001:
        p_stars = '***'
    ax = plt.gca()
    ax.annotate(p_stars, xy=(0.65, 0.6), xycoords=ax.transAxes,
                color='#b51515', fontsize=70)
sns.set(style='white', font_scale=1.6)
g = sns.PairGrid(data_rm_outlier, aspect=1.5, diag_sharey=False, despine=False)
g.map_lower(sns.regplot, lowess=True, ci=False,
            line_kws={'color': 'red', 'lw': 1},
            scatter_kws={'color': '#CF9490', 's': 20})
g.map_diag(sns.distplot, color='#809d87',
           kde_kws={'color': 'red', 'cut': 0.7, 'lw': 1},
           hist_kws={'histtype': 'bar', 'lw': 2,
                     'edgecolor': 'k', 'facecolor':'#B5CCBD'})
g.map_diag(sns.rugplot, color='#809d87')
g.map_upper(corrdot)
g.map_upper(corrfunc)
g.fig.subplots_adjust(wspace=0, hspace=0)
# Remove axis labels
for ax in g.axes.flatten():
    ax.set_ylabel('')
    ax.set_xlabel('')

# Add titles to the diagonal axes/subplots
for ax, col in zip(np.diag(g.axes),data_rm_outlier.columns):
    ax.set_title(col, y=0.82, fontsize=26)
```

```
# Split Data
X = data_rm_outlier[['Temp','pH','Tur','DO']]
y = data_rm_outlier['Class_num']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,random_state=2010)

# Data Scaling
# Min-Max Scaled
scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Fix Data Imbalance
#sm = SMOTE(kind='svm') # Actually, it have more one method (SMOTE svm method)
sm = SMOTE(kind='regular') #SMOTE regular method
X_train_balanced, y_train_balanced = sm.fit_sample(X_train_scaled, y_train)
from sklearn.metrics import classification_report
from sklearn.preprocessing import LabelBinarizer

# encode the labels as 1-hot vectors for Neural network
lb = LabelBinarizer()
y_train_oh = lb.fit_transform(y_train)
y_train_balanced_oh = lb.transform(y_train_balanced)
y_test_oh = lb.transform(y_test)

#Univariate analysis.
f = plt.figure(figsize=(40,5))
f.add_subplot(1,2,1)
# sns.distplot(data['Temp'],color='blue',axlabel="Temperature")
# sns.distplot(data['pH'],color='green',axlabel="pH")
# sns.distplot(data['Tur'],color='red',axlabel="Turbidity")
sns.distplot(data['DO'],color='purple',axlabel="Dissolved Oxygen(mg/l)")
```

```
# line plot show that data need to scaling
fig,ax = plt.subplots(figsize=(20, 5))
ax.plot(range(len(X_train)), X_train['Temp'].to_numpy(), label="Temperature")
ax.plot(range(len(X_train)), X_train['pH'].to_numpy(), label="pH")
ax.plot(range(len(X_train)), X_train['DO'].to_numpy(), label="DO")
# ax.plot(range(len(X_train)), X_train['Tur'].to_numpy(), label="Turbidity",color="red")
ax.grid(True)
ax.legend(loc='upper right')
print("Max value",X_train.max())
print("Min value",X_train.min())

# pairplot
g=sns.pairplot(data,hue="Class_num")
plt.figure(figsize = (15,10))
plt.title("Data Before SMOTE")
sns.countplot(df_y['Class_num'],palette="Set3")
# plt.title("Data After SMOTE")
# sns.countplot(y_train_balanced['Class_num'],palette="Set3")

# boxplot
plt.figure(figsize=(18,9))
data2 = data[['Temp','pH','Tur','DO']]
data2.boxplot()
plt.title("Outlier", fontsize=20)
plt.show()
```