การเปรียบเทียบความแม่นยำการจำแนกประเภทข้อมูลอนุกรมเวลาในปริภูมิเวกเตอร์ระหว่างวิธีแช็คและวิธีบอส: กรณีศึกษา ข้อมูลคลื่นไฟฟ้าหัวใจ

น.ส.นภัสสร แก้วกล้า

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาสถิติ ภาควิชาสถิติ
คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2564

THE ACCURACY COMPARISON OF TIME SERIES CLASSIFICATION IN VECTOR SPACE BETWEEN SAX AND BOSS METHODS: A CASE STUDY OF ELECTROCARDIOGRAM

Miss Napatsorn Kaewkla

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science in Statistics
Department of Statistics
FACULTY OF COMMERCE AND ACCOUNTANCY
Chulalongkorn University
Academic Year 2021

หัวข้อวิทยานิพนธ์	การเปรียบเทียบความแม่นยำการจำแนกประเภทข้อมูลอนุกรมเวลาในปริภูมิเวกเตอร์ระหว่างวิธีแซ็คและวิธีบอส: กรณีศึกษา ข้อมูลคลื่นไฟฟ้าหัวใจ
โดย	น.ส.นภัสสร แก้วกล้า
สาขาวิชา	สถิติ
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	ผู้ช่วยศาสตราจารย์ ดร.อัครินทร์ ไพบูลย์พานิช

คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต

..... คณบดีคณะพาณิชยศาสตร์และการบัญชี

(รองศาสตราจารย์ ดร.วิเลิศ ภูริวัชร)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ

(รองศาสตราจารย์ ดร.เสกสรร เกียรติสุไพบูลย์)

..... อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

(ผู้ช่วยศาสตราจารย์ ดร.อัครินทร์ ไพบูลย์พานิช)

..... กรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.นัท กุลวานิช)

..... กรรมการภายนอกมหาวิทยาลัย

(ผู้ช่วยศาสตราจารย์ ดร.สุมณฑา เกษมวิลาศ)

นภัสสร แก้วกล้า : การเปรียบเทียบความแม่นยำการจำแนกประเภทข้อมูลอนุกรมเวลาในปริภูมิเวกเตอร์ระหว่างวิธีแซ็คและวิธีบอส: กรณีศึกษา ข้อมูลคลื่นไฟฟ้าหัวใจ. (THE ACCURACY COMPARISON OF TIME SERIES CLASSIFICATION IN VECTOR SPACE BETWEEN SAX AND BOSS METHODS: A CASE STUDY OF ELECTROCARDIOGRAM) อ.ที่ปรึกษาหลัก : ผศ. ดร.อัครินทร์ ไพบูลย์พานิช

การตรวจคลื่นไฟฟ้าหัวใจ เป็นหัตถการสำคัญที่ใช้วินิจฉัยความผิดปกติของหัวใจ แต่การตรวจวัดคลื่นไฟฟ้าหัวใจนั้นก็อาจมีสัญญาณรบกวนแบบต่าง ๆ ที่เกิดขึ้นได้จากหลายสาเหตุ ซึ่งอาจทำให้ผลการวินิจฉัยทางการแพทย์ผิดพลาด งานวิจัยนี้มีวัตถุประสงค์เพื่อเปรียบเทียบอัลกอริทึมสำหรับการจำแนกประเภทข้อมูลคลื่นไฟฟ้าหัวใจที่มีสัญญาณรบกวนด้วย Symbolic Aggregate Approximation in Vector Space (SAXVSM) และ Bag of Symbolic Fourier Approximation Symbols in Vector Space (BOSSVS) เพื่อให้สามารถเลือกใช้อัลกอริทึมในการจำแนกประเภทข้อมูลคลื่นไฟฟ้าหัวใจได้อย่างเหมาะสม โดยใช้ข้อมูลคลื่นไฟฟ้าหัวใจ ECG5000 ซึ่งอยู่ในฐานข้อมูล Physionet ซึ่งข้อมูลชุดนี้ถูกบันทึกโดยศูนย์การแพทย์ Beth Israel Deaconess Medical Center (BIDMC) ที่เมืองบอสตัน ประเทศสหรัฐอเมริกา และผู้วิจัยได้จำลองการสัญญาณรบกวนในคลื่นไฟฟ้าหัวใจ 4 แบบ ได้แก่ 1) Electromyography (EMG) 2) Powerline Interference 3) Baseline Wander และ 4) Composite ที่ระดับ 25% 50% และ 100% เพื่อเปรียบเทียบประสิทธิภาพของการจำแนกประเภทจังหวะการเต้นของหัวใจปกติและผิดปกติด้วย SAXVSM และ BOSSVS จากการวิจัยสามารถสรุปได้ว่า สำหรับข้อมูลทั้ง 13 ชุด ทั้ง SAXVSM และ BOSSVSM มีประสิทธิภาพดีใกล้เคียงกัน โดยมีค่าความถูกต้องและคะแนน F1 อยู่ที่ 97-99% ค่าความแม่นยำอยู่ที่ 95-99% และค่าความระลึกอยู่ที่ 97-100% แต่ BOSSVS ใช้เวลาในการประมวลผลนานกว่า SAXVSM

สาขาวิชา สถิติ ลายมือชื่อนิสิต .....

ปีการศึกษา 2564 ลายมือชื่อ อ.ที่ปรึกษาหลัก .....

6280156126 : MAJOR STATISTICS

KEYWORD: Time Series Classification, SAX, BOSS, Vector Space, Electrocardiogram

Napatsorn Kaewkla : THE ACCURACY COMPARISON OF TIME SERIES CLASSIFICATION IN VECTOR SPACE BETWEEN SAX AND BOSS METHODS: A CASE STUDY OF ELECTROCARDIOGRAM. Advisor: Asst. Prof. AKARIN PHAIBULPANICH, Ph.D.

The electrocardiogram (ECG) is an important procedure used to diagnose heart disorders. However, the ECG may contain different types of noise due to various of factors, potentially resulting in diagnostic errors. This research compares Symbolic Aggregate Approximation in Vector Space (SAXVSM) and Bag of Symbolic Fourier Approximation Symbols in Vector Space (BOSSVS) methods for classifying ECG data with noise. To choose a suitable classification algorithm for ECG5000 dataset, which is available in the Physionet database, recorded by Beth Israel Deaconess Medical Center (BIDMC) in Boston, United States, four types of ECG noises were simulated and then added to the data as follow: 1) Electromyography (EMG) 2) Powerline Interference 3) Baseline Wander and 4) Composite at 25%, 50% and 100% levels for the performance comparison of the ECG classification between normal and abnormal heart rhythms with SAXVSM and BOSSVS. The results show that both algorithms have similar high performance for all 13 datasets: accuracy and F1 Score are 97-99%, precision is 95-99%, and recall is 97-100%, but BOSSVS has a longer running time than SAXVSM.

Field of Study: Statistics Student's Signature .....

Academic Year: 2021 Advisor's Signature .....

กิตติกรรมประกาศ

วิทยานิพนธ์เรื่อง "การเปรียบเทียบความแม่นยำการจำแนกประเภทข้อมูลอนุกรมเวลาในปริภูมิเวกเตอร์ระหว่างวิธีแซ็คและวิธีบอส: กรณีศึกษา ข้อมูลคลื่นไฟฟ้าหัวใจ " สำเร็จลุล่วงไปได้ด้วยดี ด้วยความอนุเคราะห์จากผู้มีส่วนเกี่ยวข้อง ดังต่อไปนี้

ขอขอบพระคุณผศ. ดร.อัครินทร์ ไพบูลย์พานิช อาจารย์ที่ปรึกษา ที่สละเวลาให้คำปรึกษาและติดตามความก้าวหน้าระหว่างการทำวิทยานิพนธ์นี้ ขอขอบพระคุณ รศ.ดร. เสกสรร เกียรติสุไพบูลย์ ประธานกรรมการสอบวิทยานิพนธ์ ผศ. ดร.นัท กุลวานิช และผศ.ดร.สุมณฑา เกษมวิลาศ กรรมการการสอบวิทยานิพนธ์ ที่ได้ให้คำแนะนำเพื่อให้วิทยานิพนธ์ฉบับนี้สมบูรณ์ นอกจากนี้ผู้วิจัยขอขอบคุณครอบครัวและเพื่อน ๆ ที่คอยสนับสนุนและให้คำแนะนำ จนทำให้วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยดี ผู้วิจัยขอขอบพระคุณมา ณ โอกาสนี้

นภัสสร แก้วกล้า

สารบัญ

หน้า

สารบัญตาราง

สารบัญรูปภาพ

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญ

คลื่นไฟฟ้าหัวใจ (Electrocardiogram: ECG) เป็นความต่างศักย์ของกระแสไฟฟ้าที่ไหลผ่านจุดกำเนิดไฟฟ้าไปยังเซลล์กล้ามเนื้อหัวใจทำให้หัวใจทำงาน การตรวจคลื่นไฟฟ้าหัวใจทำได้โดยนำอิเล็กโทรด (Electrode) หรือขั้วไฟฟ้าซึ่งต่อกับเครื่องบันทึกคลื่นไฟฟ้าหัวใจมาวางไว้เหนือผิวหนังบริเวณต่าง ๆ ทำให้เกิดความต่างศักย์ไฟฟ้าระหว่างส่วนต่าง ๆ ของร่างกายหรือที่เรียกว่าลีด (Lead) แต่ละลีดจะบันทึกกิจกรรมทางไฟฟ้าจากการมองหัวใจในมุมต่าง ๆ แล้วแพทย์จะวิเคราะห์รูปร่างของคลื่นแต่ละแบบที่ปรากฏบนกระดาษบันทึกคลื่นไฟฟ้าหัวใจเพื่อวินิจฉัยความผิดปกติของหัวใจและจัดความเสี่ยงต่อการเกิดโรคหัวใจประเภทต่าง ๆ (Thailand Online Hospital, 2016)

อย่างไรก็ดี ความผิดปกติของหัวใจบางอย่างก็ไม่ได้แสดงอาการ ทำให้ผู้ป่วยไม่ได้ไปพบแพทย์เพื่อรับเข้ารับรักษา ซึ่งอาจจะเป็นอันตรายต่อชีวิตได้ เช่น ภาวะหัวใจเต้นผิดจังหวะ (Cardiac Arrhythmias) คือ ภาวะที่หัวใจเต้นไม่เป็นจังหวะตามธรรมชาติ อาจเต้นเร็วหรือช้าเกินไป ทำให้หัวใจสูบฉีดเลือดไปเลี้ยงส่วนต่าง ๆ ของร่างกายไม่เพียงพอ เพิ่มความเสี่ยงต่อภาวะหัวใจล้มเหลว เกิดลิ่มเลือดไปอุดตันสมองหรืออาจทำให้เสียชีวิตได้ (ปริวัตร เพ็งแก้ว, 2563) ภาวะหัวใจเต้นผิดจังหวะเป็นภัยเงียบที่คุกคามชีวิต เนื่องจากผู้ป่วยประมาณ 15-46% จะไม่มีอาการ ขณะที่มากกว่า 65% มีอาการใจสั่น และอีกกว่า 50% มีอาการอ่อนเพลีย หายใจติดขัด หรือแน่นหน้าอก เป็นต้น ในประเทศไทยมีผู้ป่วยภาวะหัวใจเต้นผิดจังหวะจำนวนมากถึง 1.5 ล้านคน นอกจากนี้ภาวะหัวใจเต้นผิดจังหวะยังเพิ่มความเสี่ยงของการเกิดภาวะแทรกซ้อนที่อันตรายถึงชีวิต นำไปสู่ภาวะหัวใจล้มเหลว 5-6 เท่า หลอดเลือดสมองอุดตัน 2.5-3 เท่า และเสียชีวิตจากโรคหัวใจและหลอดเลือด 2-3 เท่า รวมถึงสร้างภาระในการดำเนินชีวิตของผู้ป่วย โดยพบความบกพร่องในการทำงานของร่างกาย 19% พบอุปสรรคต่อการใช้ชีวิตประจำวัน 20% ออกกำลังกายได้น้อยลงมากกว่า 50% และคุณภาพชีวิตถูกบั่นทอนมากถึง 57% (Biosense Webster, 2020)

ในการจำแนกผู้ป่วยที่มีภาวะหัวใจเต้นผิดจังหวะทำได้โดยตรวจวัดคลื่นไฟฟ้าหัวใจ อย่างไรก็ดี การตรวจวัดคลื่นไฟฟ้าหัวใจนั้นก็อาจมีปัจจัยหรือสัญญาณรบกวน (Noise) ซึ่งเกิดขึ้นได้จากหลาย

สาเหตุ เช่น ตำแหน่งของการติดเครื่องมือผิดพลาด เหงื่อ การเคลื่อนไหวร่างกาย การรบกวนของสนามแม่เหล็กไฟฟ้า เป็นต้น ซึ่งสัญญาณเหล่านี้จะเข้ามารบกวสัญญาณของคลื่นไฟฟ้าหัวใจปกติ และอาจส่งให้ผลการวินิจฉัยทางการแพทย์ผิดพลาด โดยสัญญาณรบกวนส่วนใหญ่ที่เกิดขึ้นในคลื่นไฟฟ้าหัวใจมี 4 แบบ (Chang, 2010)

1) Electromyography (EMG) เป็นสัญญาณรบกวนที่เกิดจากการหดเกร็งของกล้ามเนื้อหรือการเคลื่อนไหวของร่างกาย ซึ่งความถี่ใน EMG ทับซ้อนกันมากทำให้ตรวจจับยาก

2) Powerline Interference (50 Hz) เป็นสัญญาณรบกวนความถี่สูง โดยทั่วไปจะมีความถี่ประมาณ 50 เฮิรตซ์ สาเหตุหลักเกิดจากการรบกวนคลื่นแม่เหล็กไฟฟ้าจากสายไฟ (Electromagnetic Field *:* EMF) หรือจากเครื่องจักรในระยะใกล้ๆ การต่อสายดินที่ไม่ดี เป็นต้น

3) Baseline Wander เป็นสัญญาณรบกวนความถี่ต่ำ สาเหตุหลักเกิดจากการหายใจ การติดขั้วไฟฟ้าไม่ดี เป็นต้น

4) Composite noise เป็นสัญญาณรบกวนที่เกิดจากการเกิดร่วมกันของสัญญาณรบกวนทั้ง 3 แบบข้างต้น

ที่ผ่านมามีงานวิจัยที่ประยุกต์ใช้เครื่องมือการจำแนกประเภทข้อมูลสำหรับคลื่นไฟฟ้าหัวใจในภาวะหัวใจเต้นผิดจังหวะด้วยวิธีการที่หลากหลาย เช่น

Kaya & Pehlivan (2015) เปรียบเทียบประสิทธิภาพการจำแนกประเภทข้อมูลคลื่นไฟฟ้าหัวใจในภาวะหัวใจเต้นผิดจังหวะที่เกิดจากกระแสไฟฟ้าออกจากหัวใจห้องล่างแบบ Premature Ventricular Contraction (PVC) ใช้ข้อมูลคลื่นไฟฟ้าหัวใจจาก The MIT-BIH Arrhythmia Database แล้วสุ่มจังหวะการเต้นของหัวใจทั้งหมด 7,000 จังหวะ ประกอบด้วย ปกติ (3,500 จังหวะ) และ PVC (3,500 จังหวะ) แล้วผ่านขั้นตอนการกรองสัญญาณรบกวนด้วยค่ามัธยฐาน (Median Filtering) จากนั้นได้เปรียบเทียบวิธีการลดมิติข้อมูล (Feature Reduction) 3 วิธี คือ การวิเคราะห์องค์ประกอบหลัก (Principal Components: PCA) การวิเคราะห์องค์ประกอบอิสระ (Independent Component Analysis: ICA) และการทำแผนที่โยงก่อร่างตัวเอง (Self-Organizing Maps: SOM) แล้วจึงนำมาเปรียบเทียบประสิทธิภาพการจำแนกประเภทด้วย 4 ตัวแบบ คือ โครงข่ายประสาทเทียม (Neural Networks: NN) การหาเพื่อนบ้านที่ใกล้ที่สุด (K-Nearest Neighbour: K-NN) ซัพพอร์ตเวกเตอร์แมทชีน (Support Vector Machines: SVM)

ต้นไม้ตัดสินใจ (Decision Tree: DT) ในภาพรวมตัวแบบ K-NN ใช้เวลาในการจำแนกประเภทน้อยที่สุด ในขณะเดียวกันประสิทธิภาพการจำแนกประเภทก็ดีที่สุด ตัวแบบ K-NN ซึ่งมีแนวคิดมาจากการเปรียบเทียบข้อมูลที่สนใจกับข้อมูลอื่นว่ามีความคล้ายคลึงกันมากน้อยเพียงใด K-NN เป็นตัวแบบจำแนกประเภทที่ไม่ซับซ้อนเมื่อเทียบกับตัวแบบโครงข่ายประสาทเทียม (NN) แต่ก็สามารถให้ผลการจำแนกที่ดีกว่าในงานวิจัยของ (Kaya & Pehlivan, 2015) ได้ใช้เทคนิคการลดมิติของข้อมูลก่อนที่จะใช้ตัวแบบจำแนกประเภทเพื่อให้ใช้เวลาประมวลผลน้อยลง

Senin & Malinchik (2013) นำเสนอตัวแบบ Symbolic Aggregate Approximation - Vector Space Model (SAX-VSM) แนวคิดหลักของตัวแบบนี้ คือ จะแบ่งอนุกรมเวลาออกเป็นอนุกรมเวลาย่อย (Sliding Window) และใช้ Piecewise Aggregate Approximation (PAA) เพื่อลดมิติของข้อมูลอนุกรมเวลาก่อนแล้วจึงใช้แซ็ค (SAX) เพื่อแปลงข้อมูลอนุกรมเวลาให้เป็นรูปแบบกลุ่มตัวอักษรและใช้ตัวแบบปริภูมิเวกเตอร์ (Vector Space Model) เพื่อจำแนกประเภทข้อมูลอนุกรมเวลาจากรูปแบบกลุ่มตัวอักษรเหล่านี้ ในงานวิจัยนี้ได้เปรียบเทียบอัตราความคลาดเคลื่อนการจำแนกประเภทด้วย 5 ตัวแบบ คือ 1NN-Euclidean 1NN-Dynamic Time Wraping (DTW) Fast Shapelets Bag of Patterns และ SAX-VSM โดยใช้ข้อมูลจาก UCR Time Series ทั้งหมด 19 ชุด แสดงให้เห็นว่ามีข้อมูล 12 ชุดจากทั้งหมด 19 ชุดที่ SAX-VSM ให้อัตราความคลาดเคลื่อน (Error Rate) ต่ำกว่าตัวแบบที่เหลือ นอกจากนี้ยังเปรียบเทียบเวลาและอัตราความคลาดเคลื่อนในข้อมูล CBF Time Series ที่มีสัญญาณรบกวนระดับต่าง ๆ ระหว่างตัวแบบ 1NN-Euclidean กับ SAX-VSM แสดงให้เห็นว่า SAX-VSM ใช้เวลาประมวลผลเร็วกว่าและที่ระดับสัญญาณรบกวนเดียวกัน อัตราความคลาดเคลื่อนใน SAX-VSM ก็ต่ำกว่าตัวแบบ 1NN-Euclidean

Schäfer (2014, 2015) นำเสนอตัวแบบ Bag-Of-SFA-Symbols in Vector Space (BOSS VS) มีแนวคิดมาจากการแปลงสัมประสิทธิ์ฟูเรียร์แบบไม่ต่อเนื่อง (Discrete Fourier Transform: DFT) โดยจะกำหนดจำนวนสัมประสิทธิ์ฟูเรียร์ แล้วจึงใช้ The Multiple Coefficient Binning (MCB) เพื่อแปลงค่าสัมประสิทธิ์ฟูเรียร์นั้นให้เป็นตัวอักษร แล้วจึงใช้ Vector Space เพื่อจำแนกประเภทข้อมูลอนุกรมเวลา แล้วนำไปเปรียบเทียบประสิทธิภาพและเวลาสะสมที่ใช้ในการเรียนรู้ (Training) และทดสอบ (Testing) โดยทดลองกับข้อมูลอนุกรมเวลาจาก UCR Time Series และข้อมูลอื่น ๆ รวมทั้งหมด 91 ชุด พบว่า BOSS ใช้เวลาการประมวลผลสะสมน้อยและ

ประสิทธิภาพการจำแนกประเภทโดยเฉลี่ยดีกว่า 1NN-Dynamic Time Wraping (DTW) ในงานวิจัยระบุว่า DFT มีหลักการ คือ แปลงข้อมูลอนุกรมเวลาที่เป็นสัญญาณจากโดเมนเวลา (Time Domain) ไปเป็นโดเมนความถี่ (Frequency Domain) ทำให้สามารถลดสัญญาณรบกวนได้ และได้ทดลองเพิ่มสัญญาณรบกวนแบบเก๊าซเซี่ยล (Gaussian noise) ที่ระดับต่าง ๆ ทดสอบกับชุดข้อมูล CBF พบว่าที่ระดับสัญญาณรบกวนไม่เกิน 40% BOSS ยังคงให้ค่าความแม่นยำ (accuracy) ค่อนข้างคงที่และสูงกว่า DTW นอกจากนี้ได้ประยุกต์ใช้ BOSS ร่วมกับเทคนิคการเรียนรู้ร่วมกัน (Ensemble Classifier) โดยภาพรวมก็มีประสิทธิภาพดีเช่นกัน

จากที่กล่าวมานี้ทั้ง SAX และ BOSS มีแนวคิดสำหรับวิเคราะห์ข้อมูลอนุกรมเวลา คือ การลดมิติของอนุกรมเวลาด้วยการแบ่งอนุกรมเวลาย่อยและพยายามแปลงข้อมูลอนุกรมเวลาย่อยนี้ให้เป็นลำดับของตัวอักษรที่สามารถอธิบายได้โดยใช้ PAA และ DFT ตามลำดับ ในขณะเดียวกันก็ยังคงเก็บคุณลักษณะสำคัญของอนุกรมเวลาเดิมไว้ นอกจากนี้ยังไม่ค่อยมีผลกระทบต่อสัญญาณรบกวน แต่ยังไม่มีงานวิจัยใดใช้สองวิธีนี้เพื่อศึกษาประสิทธิภาพการจำแนกประเภทคลื่นไฟฟ้าหัวใจภาวะหัวใจห้องล่างเต้นผิดจังหวะแบบ PVC ที่มีสัญญาณรบกวนแบบต่าง ๆ ผู้วิจัยจึงมีความสนใจที่จะประยุกต์ใช้ SAX และ BOSS ร่วมกับปริภูมิเวกเตอร์ (Vector Space) ในการจำแนกประเภทข้อมูลและจำลองสัญญาณรบกวนในคลื่นไฟฟ้าหัวใจไว้ 4 แบบ คือ 1) Electromyography (EMG) 2) Powerline Interference 3) Baseline Wander และ 4) Composite Noise เพื่อเปรียบเทียบประสิทธิภาพการจำแนกประเภทของทั้งสองวิธีนี้

1.2 วัตถุประสงค์การวิจัย

เพื่อเปรียบเทียบประสิทธิภาพการจำแนกประเภทข้อมูลคลื่นไฟฟ้าหัวใจในภาวะ R-on-T Premature Ventricular Contraction (R-on-T PVC) ที่มีสัญญาณรบกวนดังต่อไปนี้

1) Electromyography (EMG)

2) Powerline Interference

3) Baseline Wander

4) Composite Noise

ที่ระดับต่าง ๆ ด้วยวิธี SAX-VSM และ BOSSVS

1.3 ขอบเขตงานวิจัย

ใช้ข้อมูลคลื่นไฟฟ้าหัวใจจาก Physionet ซึ่งเป็นแหล่งที่มาของงานวิจัยสำหรับสัญญาณทางสรีรวิทยาที่ก่อตั้งขึ้นในปี พ.ศ.2542 ภายใต้การอุปถัมภ์ของสถาบันสุขภาพแห่งชาติ (NIH) โดยมีวัตถุประสงค์เพื่อกระตุ้นให้เกิดการวิจัยด้านชีวการแพทย์ด้วยการเปิดให้เข้าถึงข้อมูลทางสรีรวิทยาการแพทย์และซอฟต์แวร์โอเพนซอร์สที่เกี่ยวข้องฟรี (MIT Laboratory for Computational Physiology, 1999)

ECG5000 เป็นหนึ่งในของข้อมูลจาก Physionet ซึ่งจะบันทึกข้อมูลคลื่นไฟฟ้าหัวใจความยาว 20 ชั่วโมงจากผู้ป่วย 1 คนที่มีภาวะหัวใจล้มเหลว จากศูนย์การแพทย์ Beth Israel Deaconess Medical Center (BIDMC) ที่เมืองบอสตัน ประเทศสหรัฐอเมริกา ในเครื่องบันทึกที่ชื่อว่า “CHf07” ข้อมูลนี้ได้ผ่านการประมวลผลมาแล้ว โดยการสุ่มคลื่นไฟฟ้าหัวใจจากจังหวะการเต้นของหัวใจในแต่ละจังหวะทั้งหมด 5,000 จังหวะ (5,000 beats) ความยาว 140 จุดเวลา (140 time points) และจัดประเภทจังหวะการเต้นของหัวใจที่ปกติและผิดปกติไว้ทั้งหมด 5 ประเภท (5 Classes) ดังตารางที่ 1

ตารางที่ 1 ลักษณะข้อมูลเบื้องต้นที่ใช้ศึกษา

ชื่อ	ประเภท	คำอธิบาย	สัดส่วน
ปกติ (Normal)	0	จังหวะการเต้นของหัวใจปกติ	0.5838
R-on-T Premature Ventricular Contraction (R-on-T PVC)	1	ภาวะที่หัวใจเต้นผิดจังหวะที่เกิดจากกระแสไฟฟ้าออกจากหัวใจห้องล่างเกิดขึ้นตรงหรือใกล้เคียงกับ T Wave ของ QRS Complexes ปกติที่นำหน้าอยู่	0.3534
Premature Ventricular Contraction (PVC)	2	ภาวะที่หัวใจเต้นผิดจังหวะที่เกิดจากกระแสไฟฟ้าออกจากหัวใจห้องล่าง	0.0388
Supraventricular Premature Beat (SP)	3	ภาวะที่หัวใจห้องบนเต้นก่อนกำหนด	0.0192
Unclassifiable Beat (UB)	4	ภาวะที่ไม่สามารถแยกจังหวะได้	0.0048

หมายเหตุ: จาก “Dataset: ECG5000” โดย Y. Chen และ E. Keogh, 2021
(http://www.timeseriesclassification.com/description.php?Dataset=ECG5000)

เนื่องจากสัดส่วนข้อมูลในประเภทที่ 2 3 และ 4 น้อยเกินไป ดังนั้นในงานวิจัยนี้จะใช้ข้อมูลในประเภท 0 คือ จังหวะการเต้นของหัวใจปกติและประเภท 1 คือ R-on-T Premature Ventricular Contraction (R-on-T PVC) จะเหลือข้อมูลทั้งหมด 4,686 จังหวะการเต้นของหัวใจ (4,686 beats) แต่ละจังหวะมีความยาว 140 จุดเวลา (140 time points) และจำลองสัญญาณรบกวนในคลื่นไฟฟ้าหัวใจ 4 แบบ ได้แก่ 1) EMG Noise 2) Powerline Interference 3) Baseline Wander และ 4) Composite Noise เพื่อเปรียบเทียบความแม่นยำในการจำแนกประเภทข้อมูลด้วย SAX-VSM และ BOSSVS

บทที่ 2

ทฤษฎีและกรอบแนวคิดที่เกี่ยวข้อง

2.1 ข้อมูลอนุกรมเวลา (Time Series Data)

ข้อมูลอนุกรมเวลา (Time series Data) คือ ข้อมูลที่ถูกเก็บรวบรวมมาตามลำดับเวลา เขียนแทนด้วย X_t โดยที่แต่ละ X คือ ค่าสังเกตที่สอดคล้องกับเวลาที่ t (ภูมิฐาน รังคกูลนุวัฒน์, 2562)

2.2 การจำแนกประเภทข้อมูลอนุกรมเวลา (Time Series Classification)

การจำแนกประเภทข้อมูลอนุกรมเวลา (Time Series Classification) คือ การประยุกต์ใช้ข้อมูลอนุกรมเวลาเพื่อจำแนกประเภทข้อมูล (Classification) ซึ่งเป็นปัญหาหนึ่งของการเรียนรู้ของเครื่องจักรแบบมีผู้สอน (Supervised machine Learning) ที่ชุดข้อมูลมีประเภทกำกับ (Class) เพื่อฝึกสอนอัลกอริทึมสำหรับจำแนกประเภทข้อมูล (av Fredrik Edin, 2020)

2.3 Symbolic Aggregate Approximation-Vector Space Model (SAX-VSM)

Symbolic Aggregate Approximation-Vector Space Model (SAX-VSM) เป็นตัวแบบหนึ่งสำหรับการจำแนกประเภทข้อมูลอนุกรมเวลา ซึ่งจะประกอบด้วย 2 ส่วน คือ Symbolic Aggregate Approximation (SAX) และ Vector Space Model (VSM) (Lin et al., 2007)

2.3.1 Symbolic Aggregate Approximation (SAX)

Symbolic Aggregate Approximation เป็นวิธีการแปลงข้อมูลอนุกรมเวลาให้เป็นรูปแบบของกลุ่มตัวอักษรเพื่อลดมิติของข้อมูล โดยมีขั้นตอนดังนี้

1) การทำข้อมูลให้เป็นมาตรฐาน (Z-normalization)

การทำข้อมูลให้เป็นมาตรฐาน (Z-normalization) คือ การแปลงหน่วยของข้อมูลทั้งหมดในชุดข้อมูลให้มีค่าเฉลี่ย 0 และส่วนเบี่ยงเบนมาตรฐาน 1 คำนวณได้ดังสมการที่ (1)

กำหนดให้ อนุกรมเวลา $T = (t_1, t_2, \ldots, t_n)$ คือ ลำดับของค่าข้อมูลที่ถูกบันทึกตามช่วงเวลา โดยที่ $n \in N$

$$T_{norm} = \frac{T-\mu}{\sigma} \qquad (1)$$

โดยที่

$T = (t_1, t_2, \ldots, t_n)$ คือ ค่าข้อมูลอนุกรมเวลา

μ คือ ค่าเฉลี่ยของอนุกรมเวลา

σ คือ ส่วนเบี่ยงเบนมาตรฐานของอนุกรมเวลา

2) การทำหน้าต่างบานเลื่อน (Sliding windows) (Schäfer, 2014)

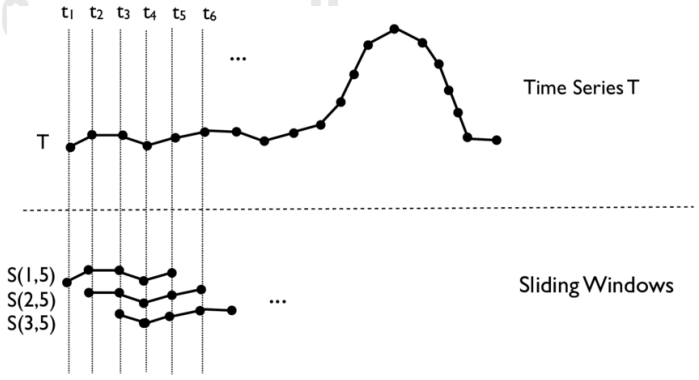
หน้าต่างบานเลื่อน (Sliding windows) คือ การแบ่งอนุกรมเวลา $T = (t_1, t_2, \ldots, t_n)$ ที่มีความยาว n ออกเป็นหน้าต่างขนาด w คงที่ จะได้ $S_{i;w} = (t_i, \ldots, t_{i+w-1})$ ดังนั้นหน้าต่างสองบานที่ติดกันจะทับซ้อนกัน $w - 1$ ตำแหน่ง แสดงได้ดังสมการที่ (2) และภาพที่ 1

$$\text{windows}(T, w) = \left\{ \underbrace{S_{1;w}}_{(t_1, t_2, \ldots, t_w)}, \underbrace{S_{2;w}}_{(t_2, t_3, \ldots, t_{w+1})}, \ldots, S_{n-w+1;w} \right\} \quad (2)$$

โดยที่

$S_{i;w}$ คือ หน้าต่างบานที่ i ความยาว w

i คือ ลำดับของหน้าต่างแต่ละบาน เมื่อ $i = 1, \ldots, n - w + 1$



ภาพที่ 1 ตัวอย่างหน้าต่างบานเลื่อน (Sliding windows)

หมายเหตุ: ตัวอย่างการใช้หน้าต่างบานเลื่อนโดยกำหนดขนาดหน้าต่างเป็น 5 (w=5)

จาก “Scalable time series similarity search for data analytics,” โดย Schäfer, 2015

3) การลดมิติของข้อมูลด้วย Piecewise Aggregate Approximation (PAA)

การลดมิติของข้อมูล (Dimension Reduction) ด้วย Piecewise Aggregate Approximation (PAA) เป็นวิธีการลดมิติของข้อมูลโดยการแบ่งข้อมูลออกเป็นส่วนย่อย ๆ p ส่วน ส่วนละเท่า ๆ กัน และคำนวณค่าเฉลี่ยของแต่ละส่วนเพื่อเป็นตัวแทนของข้อมูลในแต่ละส่วน

กำหนดให้ อนุกรมเวลามีความยาว w จุดเวลา จะแบ่งข้อมูลออกเป็น p ส่วนและหาค่าเฉลี่ยของข้อมูลในแต่ละส่วน สามารถคำนวณดังสมการที่ (3)

$$\bar{C}_i = \frac{p}{w} \sum_{j=\frac{w}{p}(i-1)+1}^{\frac{w}{p}i} C_j \qquad (3)$$

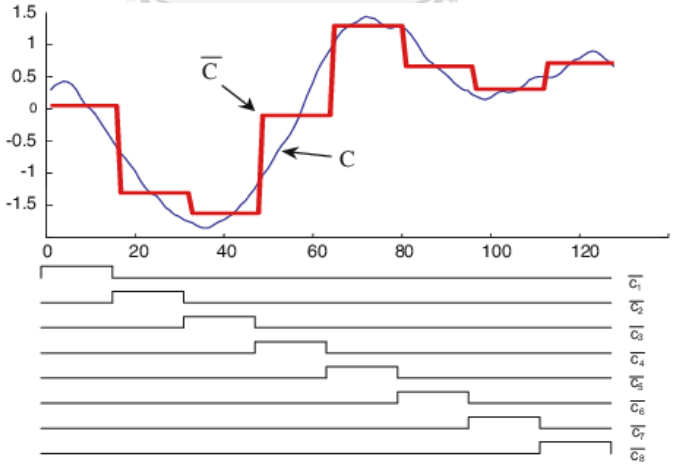
โดยที่

$i = 1, \ldots, p$

$j = 1, \ldots, w$

$\bar{C}_i$ คือ ค่าเฉลี่ยของอนุกรมเวลาส่วนที่ i

C_j คือ ค่าของข้อมูลอนุกรมเวลาจุดที่ j



ภาพที่ 2 ตัวอย่างการลดมิติข้อมูลด้วย Piecewise Aggregate Approximation (PAA)

หมายเหตุ: ตัวอย่างนี้ใช้ PAA เพื่อลดข้อมูลจาก 128 มิติเหลือ 8 มิติดังนั้นจะได้ w = 128 และ p = 8

จาก “Experiencing SAX: a novel symbolic representation of time series” โดย Lin et al., 2007

4) การแบ่งช่วงข้อมูลด้วยวิธีแซ็ค (SAX)

การแบ่งช่วงข้อมูล (Discretization) ด้วยวิธีแซ็ค (SAX) เป็นการแปลงค่าเฉลี่ยของข้อมูลในแต่ละส่วนที่ได้จากขั้นตอน PAA ให้เป็นตัวอักษร โดยจะต้องกำหนดจำนวนตัวอักษร (alphabet size) ที่ใช้ และแบ่งจำนวนช่วงของข้อมูลให้เท่ากับจำนวนตัวอักษร โดยที่พื้นที่ใต้กราฟในแต่ละช่วงจะต้องมีขนาดเท่า ๆ กัน จะได้จุดแบ่ง (Breakpoint: β) แต่ละช่วงด้วยค่ามาตรฐาน (Z- Score) ดังตารางที่ 2

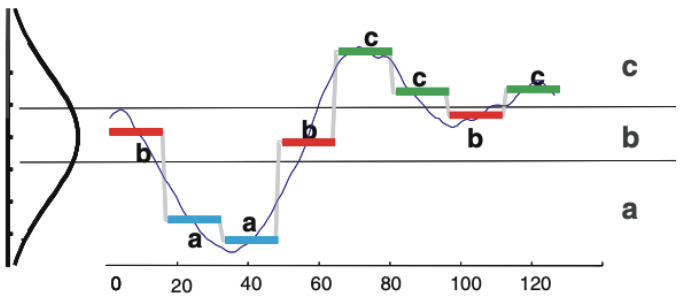
ตารางที่ 2 ตัวอย่างการกำหนดขนาดตัวอักษร (α) และจุดแบ่ง (β)

β	α							
	3	4	5	6	7	8	9	10
β_1	-0.43	-0.67	-0.84	-0.97	-1.07	-1.15	-1.22	-1.28
β_2	0.43	0	-0.25	-0.43	-0.57	-0.67	-0.76	-0.84
β_3		0.67	0.25	0	-0.18	-0.32	-0.43	-0.52
β_4			0.84	0.43	0.18	0	-0.14	-0.25
β_5				0.97	0.57	0.32	0.14	0

โดยที่

α คือ จำนวนตัวอักษรจาก 3 ถึง 10 ตัว

β_j คือ จุดแบ่งตัวที่ j สำหรับตัวอักษรต่าง ๆ เมื่อ $j = 1, \ldots, 5$



ภาพที่ 3 ตัวอย่างการแปลงข้อมูลอนุกรมเวลาโดยใช้วิธีแซ็ค (SAX)

หมายเหตุ: จากตัวอย่างจะแปลงค่าตัวแทนข้อมูลจากขั้นตอน PAA เป็นตัวอักษร โดยกำหนดความยาวของตัวอักษรเป็น 8 และจำนวนตัวอักษรเป็น 3 (A, B, C) จะได้ window size= 128 word size= 8 และ alphabet size= 3 จาก “Experiencing SAX: a novel symbolic representation of time series” โดย Lin et al., 2007

5) การลดขนาดข้อมูลด้วย Numerosity Reduction

Numerosity Reduction คือ การลดขนาดข้อมูลด้วยการเลือกใช้รูปแบบที่จำเป็นในการแสดงข้อมูล แทนการใช้ข้อมูลจริงทั้งหมด (ธรรมศักดิ์ เธียรนิเวศน์, 2548) กล่าวคือ สำหรับหน้าต่างบานเลื่อนที่อยู่ติดกัน S_{i-1}และ S_i จะแตกต่างกันเพียง 1 จุดด้านซ้ายและ 1 จุดด้านขวา ดังสมการที่ (4)

$$\text{windows}(T, w) = \left\{ \underbrace{S_{1;w}}_{(t_1, t_2, \ldots, t_w)} , \underbrace{S_{2;w}}_{(t_2, t_3, \ldots, t_{w+1})} , \ldots , S_{n-w+1;w} \right\} \qquad (4)$$

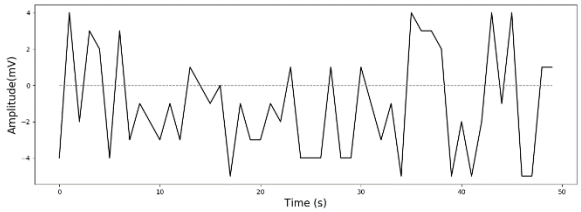
โดยที่

$S_{i;w}$ คือ หน้าต่างบานที่ i ความยาว w

i คือ ลำดับของหน้าต่างแต่ละบาน เมื่อ $i = 1, \ldots, n - w + 1$

ดังนั้น หน้าต่างบานเลื่อนที่อยู่ติดกัน S_{i-1}และ S_i มีแนวโน้มที่จะได้รูปแบบตัวอักษร (pattern) ชุดเดียวกัน เพื่อหลีกเลี่ยงการนับจำนวนของรูปแบบของตัวอักษรที่ซ้ำกันมากเกินไป จะใช้เทคนิค numerosity reduction โดยจะนับรูปแบบของตัวอักษรที่ซ้ำกันในหน้าต่างบานเลื่อนที่ติดกันเพียงครั้งเดียวและจะนับซ้ำอีกครั้ง ก็ต่อเมื่อ พบรูปแบบของตัวอักษรเดิมที่เกิดขึ้นอีกในหน้าต่างลำดับอื่น ๆ ที่ไม่ติดกัน (Lin et al., 2007)

(1) การทำข้อมูลให้เป็นมาตรฐาน (Z-normalization)



(2) การทำหน้าต่างบานเลื่อน (Sliding windows)

$$\text{windows}(T, w) = \left\{ \underbrace{S_{1;w}}_{(t_1, t_2, \ldots, t_w)} , \underbrace{S_{2;w}}_{(t_2, t_3, \ldots, t_{w+1})} , \ldots , S_{n-w+1;w} \right\}$$

(3) การแปลงฟูเรียร์เชิงสัญลักษณ์ (Symbolic Aggregate Approximation: SAX)

$$S_{1;w} = (\text{aaccbabc}, \text{aaccbabc}, \text{aaccbabc}, \ldots \text{baccbcaa}, \text{bbcbaaca}, \text{ccaccbaa})$$

$$S_{2;w} = (\text{baccbabc}, \text{acccccbc}, \text{bbccbabc}, \ldots \text{bacccaba}, \text{bacccaba}, \text{ccaccbaa})$$

$$\vdots$$

$$: S_{n-w+1;w} = (\text{baccbbaa}, \text{cbcabbcc}, \text{cbcabbcc}, \ldots \text{abccaacc}, \text{acccbabc}, \text{bbcbaaca})$$

(4) การลดขนาดข้อมูลด้วย Numerosity Reduction

$$S_{1;w} = (\text{aaccbabc}, \text{aaccbabc}, \ldots \text{baccbcaa}, \text{bbcbaaca}, \text{ccaccbaa})$$

$$S_{2;w} = (\text{baccbabc}, \text{acccccbc}, \text{bbccbabc}, \ldots \text{bacccaba}, \text{ccaccbaa})$$

$$\vdots$$

$$S_{n-w+1;w} = (\text{baccbbaa}, \text{cbcabbcc}, \ldots \text{abccaacc}, \text{acccbabc}, \text{bbcbaaca})$$

(5) ความถี่ของรูปแบบตัวอักษรต่าง ๆ Bag Of SAX Symbol: SAX

Window	aaccbabc	abcceaaa	bbcceaaa	babccaaa	abbcabbc	...
1	0	10	3	1	0	...
2	1	0	2	11	1	...
:						
n	...	...	...	...	...	...

ภาพที่ 4 สรุปขั้นตอนของ Symbolic Aggregate Approximation (SAX)

2.3.2 แบบจำลองปริภูมิเวกเตอร์ (Vector Space Model: VSM)

แบบจำลองปริภูมิเวกเตอร์ (Vector Space Model: VSM) เป็นวิธีหนึ่งในการแทนเอกสารที่ไม่มีโครงสร้าง (Unstructured Text Document) ด้วยแบบจำลองปริภูมิเวกเตอร์ โดยกำหนดให้เอกสารแต่ละฉบับเปรียบเสมือนเวกเตอร์ของคำ ขนาดของเวกเตอร์ขึ้นอยู่กับจำนวนของคำที่ปรากฏอยู่ในเอกสารฉบับนั้น

โดยที่

W_{ik} คือ น้ำหนักของคำ K ที่ปรากฏในเอกสารฉบับที่ i

D_i คือ เวกเตอร์สำหรับเอกสาร เขียนแทนด้วย $D_i = (W_{i1}, W_{i2}, \ldots, W_{it})$

t คือ จำนวนของคำที่ไม่ซ้ำกันในชุดของเอกสารทั้งหมด ดังนั้นในช่องว่าง (Space) ของเอกสารชุดหนึ่งจะมีมิติเท่ากับ t -มิติ (Raghavan & Wong, 1986)

1) การให้น้ำหนักคำ (Term Weighting)

การให้น้ำหนักคำ (Term Weighting) เป็นวิธีการให้น้ำหนักสำหรับคำที่มีความสำคัญหรือใช้เป็นตัวแทนของเอกสารที่ควรจะปรากฏอยู่เป็นจำนวนมากในเนื้อหาของเอกสารเฉพาะฉบับนั้นและปรากฏอยู่น้อยในชุดของเอกสารที่เหลือทั้งหมด แต่ถ้าคำนั้นปรากฏเป็นจำนวนมากในทุกเอกสารแสดงว่าคำดังกล่าวไม่สามารถเป็นตัวแทนของเอกสารใด ๆ ได้ ซึ่งคำเหล่านั้นเรียกว่าคำหยุด (Stop Word) ดังนั้นการให้น้ำหนักคำ ๆ หนึ่งในเอกสารฉบับหนึ่งจะพิจารณาจากความถี่ของคำ (Term Frequency) ที่ปรากฏในเอกสารนั้นและจำนวนของเอกสารทั้งหมดที่มีคำ ๆ นั้นปรากฏอยู่ วิธีการให้น้ำหนักของคำที่จะใช้ในงานวิจัยนี้ คือ การหาความถี่ของคำและการผกผันความถี่ในเอกสาร TF-IDF (Salton & Buckley, 1988)

การหาความถี่ของคำและการผกผันความถี่ในเอกสาร (Term Frequency-Inverse Document Frequency: TF-IDF) เป็นวิธีการทางสถิติวิธีหนึ่งในการประเมินความสำคัญของคำต่อเอกสาร ซึ่งความสำคัญของคำจะเป็นสัดส่วนโดยตรงกับจำนวนครั้งที่คำคำนั้นปรากฏในเอกสารนั้น แต่จะถูกลดความสำคัญลงด้วยความถี่ของคำนั้นในกลุ่มเอกสารทั้งหมด (Schäfer, 2015b) โดยมีรายละเอียดดังนี้

$tf_{t,T}$ ใช้สำหรับหาค่าความถี่ของคำ (t) ในอนุกรมเวลา T แบบมาตราส่วนลอการิทึม (Logarithmic scale) สามารถคำนวณได้ดังสมการที่ (5)

$$\mathrm{tf}_{t,T} = \begin{cases} 1 + \log\left(\mathrm{B}_T(t)\right) & \text{ถ้า } \mathrm{B}_T(t) > 0 \\ 0 & \text{กรณีอื่น ๆ} \end{cases} \quad (5)$$

โดยที่

T คือ อนุกรมเวลา

$\mathrm{B}_T(t)$ คือ ความถี่ของคำ (t) ของอนุกรมเวลา T

$\mathrm{tf}_{t,C}$ ใช้หาค่าความถี่ของคำ (t) ในประเภท (C) แบบมาตราส่วนลอการิทึม (Logarithmic scale) สามารถคำนวณได้ดังสมการที่ (6)

$$\mathrm{tf}_{t,C} = \begin{cases} 1 + \log(\sum_{T \epsilon C} \mathrm{B}_T(t)) & \text{ถ้า } \sum_{T \epsilon C} \mathrm{B}_T(t) > 0 \\ 0 & \text{กรณีอื่น ๆ} \end{cases} \quad (6)$$

โดยที่

$\sum_{T \epsilon C} \mathrm{B}_T(t)$ คือ ความถี่ของคำ (t) ในอนุกรมเวลา T ที่อยู่ในประเภท (C)

$\mathrm{idf}_{t,C}$ ใช้สำหรับวัดความสำคัญของคำ (t) ในประเภท (C) ทั้งหมด คำนวณได้ดังสมการที่ (7)

$$\mathrm{idf}_{t,C} = \log\left(1 + \frac{|\mathrm{CLASSES}|}{|\{C \mid T \in C \cap \mathrm{B}_T(t) > 0\}|}\right) \quad (7)$$

โดยที่

$|\{C \mid T \in C \cap \mathrm{B}_T(t) > 0\}|$ คือ จำนวนประเภท (C) ที่พบคำ (t)

$|\mathrm{CLASSES}|$ คือ จำนวนประเภท (C) ทั้งหมด

$\mathrm{tf} * \mathrm{idf}(t, C)$ ใช้วัดความสำคัญของคำ (t) ในประเภท (C) แต่จะถูกลดทอนด้วยความสำคัญของคำ (t) ในประเภท (C) ทั้งหมด สามารถคำนวณได้ดังสมการที่ (8)

$$\mathrm{tf} * \mathrm{idf}(t, C) = 1 + \log(\textstyle\sum_{T \epsilon C} B_T(t)) \times \log\left(1 + \frac{|\mathrm{CLASSES}|}{|\{C \mid T \epsilon C \cap B_T(t) > 0\}|}\right) \quad (8)$$

ตารางที่ 3 ตัวอย่าง TF-IDF

Term	Class1	Class2	...	Class n
aabbcba	$\mathrm{tf} * \mathrm{idf}_{1,1}$	$\mathrm{tf} * \mathrm{idf}_{1,2}$	...	$\mathrm{tf} * \mathrm{idf}_{1,n}$
ccccbbaa	$\mathrm{tf} * \mathrm{idf}_{2,1}$	$\mathrm{tf} * \mathrm{idf}_{2,2}$	...	$\mathrm{tf} * \mathrm{idf}_{2,n}$
baccaacc	$\mathrm{tf} * \mathrm{idf}_{3,1}$	$\mathrm{tf} * \mathrm{idf}_{3,2}$	...	$\mathrm{tf} * \mathrm{idf}_{3,n}$
...	...	...	...	...

2) วิธีวัดความเหมือน (Similarity Measure)

วิธีวัดความเหมือน (Similarity Measure) คือ วิธีการทางสถิติหนึ่งที่ใช้วัดค่าความคล้ายคลึงระหว่างเอกสารกับคลังข้อมูล ในงานวิจัยนี้จะใช้ค่าความเหมือนโคไซน์ (Cosine similarity)

ค่าความเหมือนโคไซน์ (Cosine similarity) คือ การหาความคล้ายคลึงด้วยองศา เป็นการวัดความเหมือนของเวกเตอร์ 2 เวกเตอร์ว่าไปในทิศทางเดียวกันหรือไม่ โดยตัดขนาด (Magnitude) ของเวกเตอร์ออกไป (Srikong, 2020) สามารถคำนวณได้ดังสมการที่ (9) และผลลัพธ์ของอนุกรมเวลาพิจารณาจากประเภทหรือ Class ที่ให้ค่าความเหมือนโคไซน์ที่มากที่สุด ดังสมการที่ (10)

$$\mathrm{similarity}(Q, C) = \frac{\vec{Q} \cdot \vec{C}}{||\vec{Q}|| \cdot ||\vec{C}||} = \frac{\sum_{t \epsilon Q} \mathrm{tf}(t,Q) \cdot (\mathrm{tf} * \mathrm{idf}(t,C) + 1)}{\sqrt{\sum_{t \epsilon Q} (\mathrm{tf}(t,Q))^2} \sqrt{\sum_{t \epsilon C} (\mathrm{tf} * \mathrm{idf}(t,C))^2}} \quad (9)$$

$$\mathrm{label}(Q) = \underset{C \in \mathrm{CLASSES}}{\mathrm{argmax}} \left(\mathrm{similarity}(Q, C)\right) \quad (10)$$

โดยที่

$\text{similarity}\ (Q, C)$ คือ ค่าความเหมือนโคไซน์ของอนุกรมเวลา Q กับประเภท C

$\text{label}(Q)$ คือ ผลลัพธ์หรือประเภทของอนุกรมเวลา Q

$\sum_{t \epsilon Q} \text{tf}(t, Q)$ คือ ความถี่ของคำ (t) ในอนุกรมเวลา Q

$\text{tf} * \text{idf}\ (t, C)$ คือ ความถี่ของคำและการผกผันความถี่ (TF-IDF) ของคำในประเภท C

2.4 Bag of Symbolic Fourier Approximation Symbols-Vector Space (BOSS VS)

Bag of Symbolic Fourier Approximation Symbols-Vector Space (BOSS VS) เป็นตัวแบบหนึ่งสำหรับการจำแนกประเภทข้อมูลอนุกรมเวลา ซึ่งจะประกอบด้วย 2 ส่วน คือ Bag of Symbolic Fourier Approximation Symbols (BOSS) และ Vector Space (VS) (Schäfer, 2015a)

2.4.1 การประมาณฟูเรียร์เชิงสัญลักษณ์ (Symbolic Fourier Approximation: SFA)

การประมาณฟูเรียร์เชิงสัญลักษณ์ Symbolic Fourier Approximation (SFA) คือ วิธีหนึ่งในการแปลงข้อมูลอนุกรมเวลาให้เป็นรูปแบบตัวอักษร โดยมี 2 ขั้นตอนแรก คือ การทำข้อมูลให้เป็นมาตรฐาน (Z-normalization) และการทำหน้าต่างบานเลื่อน (Sliding windows) ตามที่ได้กล่าวมาในขั้นตอนของแซ็ค (SAX) ซึ่งอยู่ในหัวข้อ 2.3.1 ข้อที่ 1) และ 2) ตามลำดับ และมีขั้นตอนต่อไปดังนี้

1) การประมาณค่าสัมประสิทธิ์การแปลงฟูเรียร์แบบไม่ต่อเนื่อง (DFT)

การประมาณค่า (Approximation) การแปลงฟูเรียร์แบบไม่ต่อเนื่อง (Discrete Fourier Transform: DFT) เป็นการประมาณค่าสัมประสิทธิ์ของอนุกรมฟูเรียร์จากข้อมูลสัญญาณรายจุดหรือสัญญาณไม่ต่อเนื่อง (Discrete) รายคาบตั้งแต่ 0 ถึง N-1 ที่แบ่งมาจากสัญญาณต่อเนื่อง เพื่อที่จะแปลงสัญญาณโดเมนเวลาให้เป็นโดเมนความถี่ (Souspace, 2020) สามารถคำนวณได้ดังสมการที่ (11)

$$X(m) = \sum_{n=0}^{N-1} x(n) e^{\frac{-j2\pi nm}{N}} \quad , \ j = \sqrt{-1} \tag{11}$$

โดยที่

$n, m = 0, 1, 2, .., N-1$

$x(n)$ คือ ลำดับของตัวเลขจากสัญญาณไม่ต่อเนื่อง

$X(m)$ คือ ผลการแปลงฟูเรียร์แบบไม่ต่อเนื่อง (DFT) ของ $x(n)$

จากสมการของออยเลอร์ (Euler's Equation)

$$e^{j\theta} = \cos(\theta) - j\sin(\theta) \qquad (12)$$

ถ้าแทน $\theta = \frac{2\pi nm}{N}$ ในสมการที่ (12) จะได้

$$e^{\frac{-j2\pi nm}{N}} = \cos\left(\frac{2\pi nm}{N}\right) - j\sin\left(\frac{2\pi nm}{N}\right) \qquad (13)$$

และแทนสมการที่ (13) ในสมการที่ (11) จะได้

$$X(m) = \sum_{n=0}^{N-1} x(n)\left[\cos\left(\frac{2\pi nm}{N}\right) - j\sin\left(\frac{2\pi nm}{N}\right)\right] \qquad (14)$$

โดยที่

$\cos(\frac{2\pi nm}{N}) - j\sin(\frac{2\pi nm}{N})$ คือ ตัวแทนของสัญญาณแบบไซด์ความถี่ f_mที่ จุด n

จะได้ว่า $X(m)$ เป็นค่าสัมประสิทธิ์การแปลงฟูเรียร์ ซึ่งจะอยู่ในรูปของจำนวนเชิงซ้อน ประกอบด้วย (ส่วนจริง(m) , ส่วนจินตภาพ(m)) โดยที่ $m = 0, 1, 2, .., N-1$

2) การทำควอนไทซ์ (Quantisation)

การทำควอนไทซ์ (Quantisation) เป็นการกำหนดช่วงของข้อมูลที่จะแปลงค่าประมาณสัมประสิทธิ์การแปลงฟูเรียร์แบบไม่ต่อเนื่อง (DFT) เป็นตัวอักษร (alphabet) ด้วยเทคนิค The Multiple Coefficient Binning (MCB) โดยมีการเตรียมข้อมูลดังนี้

กำหนดให้ เมทริกซ์ A ที่มาจากขั้นตอนการแปลงฟูเรียร์แบบไม่ต่อเนื่อง (DFT) ของชุดข้อมูลเรียนรู้ (Training Data) แสดงได้สมการที่ (15)

$$A = \begin{pmatrix} DFT(T_1) \\ . \\ DFT(T_i) \\ . \\ DFT(T_N) \end{pmatrix} = \begin{pmatrix} real_{1,1} & img_{1,1} & \dots & real_{1,\frac{l}{2}} & img_{1,\frac{l}{2}} \\ . & . & \dots & . & . \\ real_{i,1} & img_{i,1} & \dots & real_{i,\frac{l}{2}} & img_{i,\frac{l}{2}} \\ . & . & \dots & . & . \\ real_{N,1} & img_{N,1} & \dots & real_{N,\frac{l}{2}} & img_{N,\frac{l}{2}} \end{pmatrix} = (C_1, C_2, \dots, C_l) \qquad (15)$$

โดยที่

$i = 1, \ldots, N$

$k = 1, .., \frac{l}{2}$

$j = 1, \ldots, l$

T_i	คือ	ชุดข้อมูลเรียนรู้ (Training Data) แถวที่ i
$real_{ik}$	คือ	ค่าจริงของค่าสัมประสิทธิ์ฟูเรียร์ตัวที่ k ของข้อมูลเรียนรู้ตัวที่ i
img_{ik}	คือ	ค่าจินตภาพของค่าสัมประสิทธิ์ฟูเรียร์ตัวที่ k ของข้อมูลเรียนรู้ตัวที่ i
C_j	คือ	ค่าสัมประสิทธิ์ฟูเรียร์คอลัมน์ที่ j

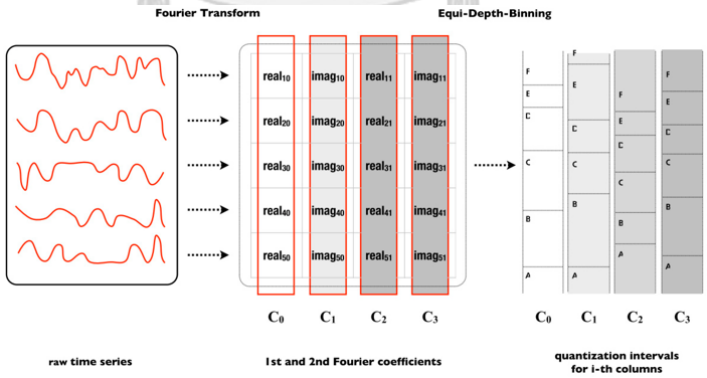
The Multiple Coefficient Binning (MCB) และมีขั้นตอนดังนี้

ขั้นตอนที่ 1 เรียงค่าของข้อมูลในแต่ละคอลัมน์ C_j จากเมทริกซ์ A

ขั้นตอนที่ 2 กำหนดจำนวนตัวอักษรเท่ากับ a และแบ่งข้อมูลออกเป็น a ส่วน โดยที่จำนวนข้อมูลในแต่ละส่วนเท่า ๆ กัน (equi-depth binning bins) ดังนั้นแต่ละคอลัมน์ C_j จะมีจุดแบ่ง (breakpoints) แตกต่างกัน

ขั้นตอนที่ 3 กำหนดลำดับตัวอักษรประจำตำแหน่งในแต่ละส่วนในคอลัมน์ C_j

จะได้ตัวอย่างการทำ The Multiple Coefficient Binning (MCB) แสดงได้ดังภาพที่ 5

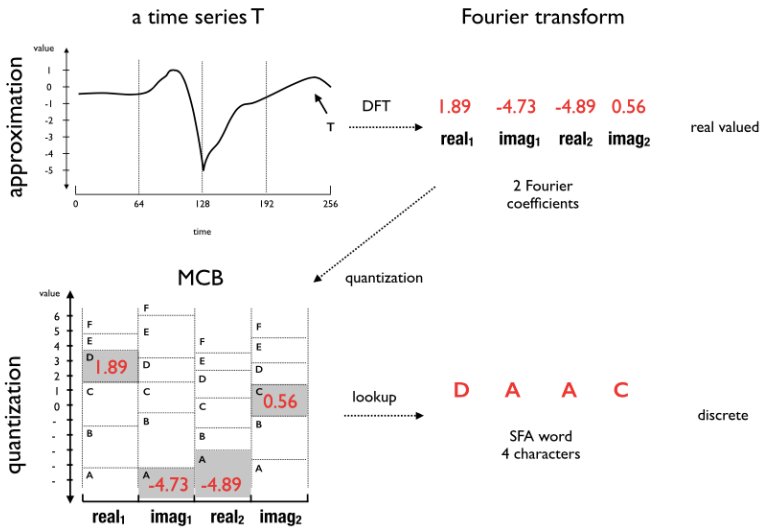


ภาพที่ 5 ขั้นตอน The Multiple Coefficient Binning (MCB)

หมายเหตุ: การแปลงค่าข้อมูลอนุกรมเวลาโดยใช้ DFT และเรียงค่าข้อมูลในคอลัมน์ C_j และแบ่งข้อมูลในแต่ละส่วนเท่า ๆ กัน จากตัวอย่างกำหนด j = 4 และ กำหนด a = 6 (A,B,C,D,E,F)

จาก “Human Activity Recognition Based on Symbolic Representation Algorithms for Inertial Sensors” โดย Wesllen Sousa Lima et al., 2018

และตัวอย่างขั้นตอนการแปลงฟูเรียร์เชิงสัญลักษณ์ (SFA) แสดงได้ดังภาพที่ 6



ภาพที่ 6 ตัวอย่างขั้นตอนของการแปลงฟูเรียร์เชิงสัญลักษณ์ (SFA)

หมายเหตุ: (1) การประมาณ (Approximation) ด้วย DFT = (1.89, -4.73, -4.89, 0.56)

(2) การทำควอนไทซ์ (Quantisation) ด้วย MCB โดยกำหนด a = 6 (A, B, C, D, E, F)

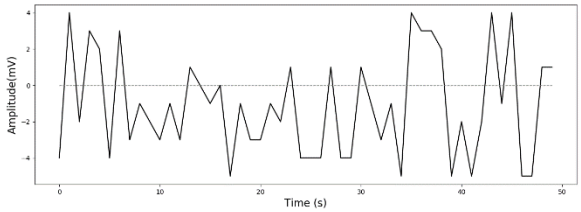
(3) ตัวอักษรจากฟูเรียร์ คือ DAAC

จาก “Bag-Of-SFA-Symbols in Vector Space (BOSS VS)” โดย Schäfer, 2015

3) Bag Of SFA Symbol: BOSS

Bag Of SFA Symbol: BOSS เป็นการหาความถี่ของรูปแบบตัวอักษรฟูเรียร์เชิงสัญลักษณ์ทั้งหมด (SFA Word) และลดขนาดข้อมูลโดยใช้ Numerosity Reduction ตามที่ได้กล่าวมาแล้วในหัวข้อ 2.3.1 ข้อที่ 5) และสามารถสรุปขั้นตอนวิธี Bag Of SFA Symbol (BOSS) ได้ดังภาพที่ 7

(1) การทำข้อมูลให้เป็นมาตรฐาน (Z-normalization)



(2) การทำหน้าต่างบานเลื่อน (Sliding windows)

$$\text{windows}(T, w) = \left\{ \underbrace{S_{1;w}}_{(t_1, t_2, \ldots, t_w)}, \underbrace{S_{2;w}}_{(t_2, t_3, \ldots, t_{w+1})}, \ldots, S_{n-w+1;w} \right\}$$

(3) การแปลงฟูเรียร์เชิงสัญลักษณ์ (Bag of Symbolic Fourier Approximation: BOSS)

$S_{1;w} = (\text{aaccbabc}, \text{aaccbabc}, \text{aaccbabc}, \ldots \text{baccbcaa}, \text{bbcbaaca}, \text{ccaccbaa})$

$S_{2;w} = (\text{baccbabc}, \text{acccccbc}, \text{bbccbabc}, \ldots \text{bacccaba}, \text{bacccaba}, \text{ccaccbaa})$

:

$: S_{n-w+1;w} = (\text{baccbbaa}, \text{cbcabbcc}, \text{cbcabbcc}, \ldots \text{abccaacc}, \text{acccbabc}, \text{bbcbaaca})$

(4) การลดขนาดข้อมูลด้วย Numerosity Reduction

$S_{1;w} = (\text{aaccbabc}, \text{aaccbabc}, \ldots \text{baccbcaa}, \text{bbcbaaca}, \text{ccaccbaa})$

$S_{2;w} = (\text{baccbabc}, \text{acccccbc}, \text{bbccbabc}, \ldots \text{bacccaba}, \text{ccaccbaa})$

:

$S_{n-w+1;w} = (\text{baccbbaa}, \text{cbcabbcc}, \ldots \text{abccaacc}, \text{acccbabc}, \text{bbcbaaca})$

(5) ความถี่ของรูปแบบตัวอักษรต่าง ๆ Bag Of SFA Symbol: BOSS

Window	aaccbabc	abcceaaa	bbcceaaa	babccaaa	abbcabbc	...
1	0	10	3	1	0	...
2	1	0	2	11	1	...
:						
n	...	...	...	...	...	...

ภาพที่ 7 สรุปขั้นตอนของ Bag of Symbolic Fourier Approximation (BOSS)

2.5 วิธีการตรวจสอบไขว้ (K-Fold Cross Validation)

วิธีการตรวจสอบไขว้ (K-Fold Cross Validation) คือ การแบ่งข้อมูลออกเป็นหลาย ๆ ส่วน ส่วนละเท่า ๆ กัน จำนวน K ส่วน (K Fold) โดยให้ส่วนหนึ่งเป็นชุดข้อมูลทดสอบ (Testing Data) และส่วนที่เหลือเป็นชุดข้อมูลเรียนรู้ (Training Data) แล้วทำการทดสอบวนไปเรื่อย ๆ K รอบ แต่ละรอบสลับให้ข้อมูลแต่ละส่วนได้เป็นข้อมูลชุดทดสอบ (Testing Data) (ขนิษฐา ดีสุบิน, 2560) ดังตารางที่ 4 เป็นการแบ่งข้อมูลออกเป็น 5 ส่วน แต่ละส่วนจะมีข้อมูลเรียนรู้ (70%) และข้อมูลทดสอบ (30%)

ตารางที่ 4 วิธีการตรวจสอบไขว้ (K-Fold Cross Validation)

Fold	Training Data (100%) K-Fold Cross Validation				
1	Train (70%)				Test (30%)
2	Train (70%)			Test (30%)	Train (70%)
3	Train (70%)		Test (30%)	Train (70%)	
4		Test (30%)	Train (70%)		
5	Test (30%)	Train (70%)			

2.6 การวัดประสิทธิภาพ (Performance Evaluation)

Confusion Matrix คือ เมทริกซ์ที่แสดงผลลัพธ์จากการทำนาย (Prediction) โดยแสดงรายละเอียดการแบ่งประเภทข้อมูล (Gatchalee, 2019) ดังตารางที่ 5

ตารางที่ 5 Confusion Matrix

	ค่าทำนาย (Predicted Class)	
ค่าจริง (Actual Class)	**Positive**	**Negative**
Positive	True Positive (TP)	False Negative (FN)
Negative	False Positive (FP)	True Negative (TN)

True Positive (TP) คือ จำนวนข้อมูลที่ถูกทำนายว่า “จริง” และ มีค่าเป็น “จริง”

True Negative (TN) คือ จำนวนข้อมูลที่ถูกทำนายว่า “ไม่จริง” และ มีค่า “ไม่จริง”

False Positive (FP) คือ จำนวนข้อมูลที่ถูกทำนายว่า “จริง” แต่ มีค่าเป็น “ไม่จริง”

False Negative (FN) คือ จำนวนข้อมูลที่ถูกทำนายว่า “ไม่จริง” แต่ มีค่าเป็น “จริง”

ค่าความถูกต้อง (Accuracy) สามารถคำนวณได้ดังสมการที่ (16)

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}} \tag{16}$$

ค่าความแม่นยำ (Precision) สามารถคำนวณได้ดังสมการที่ (17)

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \tag{17}$$

ค่าความระลึก (Recall) สามารถคำนวณได้ดังสมการที่ (18)

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}} \tag{18}$$

คะแนน F1 (F1 Score) สามารถคำนวณได้ดังสมการที่ (19)

$$\text{F1 Score} = \frac{2 \times (\text{Precision} \times \text{Recall})}{\text{Precsion+Recall}} \tag{19}$$

2.7 กรณีศึกษา: ข้อมูลคลื่นไฟฟ้าหัวใจ (Electrocardiography: ECG)

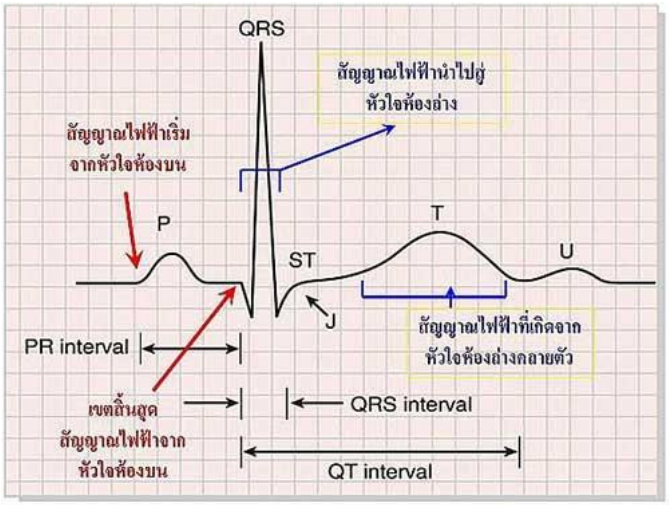
คลื่นไฟฟ้าหัวใจ (Electrocardiography: ECG) เป็นการทดสอบสัญญาณไฟฟ้าของหัวใจในแต่ละจังหวะการเต้นของหัวใจ ซึ่งจะถูกปล่อยออกมาและส่งผ่านไปทั่วทั้งหัวใจ ส่งผลให้กล้ามเนื้อหัวใจมีการบีบตัวอย่างสมบูรณ์ในการส่งเลือดไปเลี้ยงส่วนต่าง ๆ ของร่างกาย (Medthai, 2018)

การตรวจคลื่นไฟฟ้าหัวใจ เป็นการตรวจการทำงานของหัวใจเมื่อกล้ามเนื้อหัวใจบีบตัว (systole) และคลายตัว (diastole) สลับกันไปเป็นจังหวะอย่างสม่ำเสมอ ซึ่งเป็นการทำงานของเซลล์ในหัวใจเกิดเป็นระบบการนำไฟฟ้าของหัวใจ (conduction system)

การบันทึกคลื่นไฟฟ้าหัวใจ เป็นการบันทึกกราฟกระแสไฟฟ้าที่ผลิตขึ้นจากหัวใจในทุกทิศทางไปถึงผิวหนัง สามารถวัดได้โดยใช้แผ่นโลหะซึ่งเป็นอิเล็กโทรดที่ต่อกับเครื่องบันทึกคลื่นไฟฟ้าหัวใจแล้วต่อเข้ากับร่างกายของผู้ป่วย เพื่อบันทึกคลื่นไฟฟ้าหัวใจจากความต่างศักย์ไฟฟ้าที่เกิดขึ้นจากขั้วไฟฟ้าทั้งขั้วบวกและขั้วลบที่พื้นผิวของร่างกาย หรือที่เรียกว่า ลีด (lead) แต่ละลีดจะบันทึกกิจกรรมทางไฟฟ้าของหัวใจในมุมต่าง ๆ เช่น หากวางขั้วไฟฟ้าไว้ที่แขนขา จะเรียกว่า ลิมบ์ลีด (limb leads) โดยตำแหน่งการวางขั้วไฟฟ้าเพื่อบันทึกคลื่นไฟฟ้าหัวใจจะแตกต่างกันขึ้นอยู่กับชนิดของขั้วไฟฟ้า ที่นิยมมี 12 ขั้วไฟฟ้าหรือ 12 ลีด (Thailand Online Hospital, 2016)

แม้ว่าการวินิจฉัยข้อมูลเชิงลึกของสภาวะของหัวใจทำได้โดยการตรวจสอบคลื่นไฟฟ้าหัวใจจากทุก ๆ ลีด เพราะจะเห็นลักษณะคลื่นในหลายเวกเตอร์ไฟฟ้าเพื่อประโยชน์ในการวินิจฉัย แต่การใช้เพียงลีดเดียวที่เรียกว่าแถบจังหวะ ก็สามารถนำมาใช้ดูแนวโน้มการทำงานของหัวใจขั้นพื้นฐานและตรวจหาภาวะหัวใจเต้นผิดจังหวะแบบต่าง ๆ ได้ (Pathangay & Rath, 2014)

2.7.1 ส่วนประกอบของคลื่นไฟฟ้าหัวใจ



ภาพที่ 8 ส่วนประกอบของคลื่นไฟฟ้าหัวใจ
หมายเหตุ: จาก “คลื่นไฟฟ้าหัวใจ” โดย Wut Wuttipong
(http://wuttipongwut.blogspot.com/p/blog-page_9.html)

1) P wave เกิดจากปุ่มหัวใจห้องบนเอสเอโนด (SA node) ส่งกระแสมากระตุ้นที่หัวใจห้องบนขวาและห้องบนซ้ายเกิดการดีโพลาไรเซชัน (Depolarization)

2) PR interval เป็นระยะทางที่เริ่มจากเอสเอโนด (SA node) ส่งคลื่นไฟฟ้ามาเกิดการดีโพลาไรเซชัน (Depolarization) ที่หัวใจห้องบน จากนั้นลงสู่ปุ่มเอวีโนด (AV node) ที่หัวใจห้องล่าง

3) QRS complex เป็นผลรวมทางไฟฟ้าจากการดีโพลาไรเซชัน (Depolarization) ของหัวใจห้องล่างซ้ายและขวา

4) T wave เป็นผลรวมทางไฟฟ้าจากการรีโพลาไรเซชัน (Repolarization) ของหัวใจห้องล่าง สำหรับการรีโพลาไรเซชัน (Repolarization) ของหัวใจห้องบนมักมองไม่เห็น เพราะเกิดในช่วงการดีโพลาไรเซชัน (Depolarization) ของหัวใจห้องล่าง

5) ST segment เป็นการเริ่มการรีโพลาไรเซชัน (Repolarization) ของหัวใจห้องล่าง นับจากจุดสิ้นสุดของ QRS complex จุดต่อตรงนี้เรียกว่า j point ไปจนถึงจุดเริ่มต้น T wave

6) QT interval เป็นระยะเวลารวมของทั้งการดีโพลาไรเซชัน (Depolarization) และการรีโพลาไรเซชัน (Repolarization) รวมกัน วัดตั้งแต่เริ่ม QRS complex ไปจนสิ้นสุด T wave

7) U wave มีขนาดเล็กโค้งกลมตามหลัง T wave ส่วนใหญ่เห็นชัดใน lead V2 – V3 มีทิศทางไปในทางเดียวกับ T wave

2.7.2 สัญญาณรบกวนในคลื่นไฟฟ้าหัวใจ

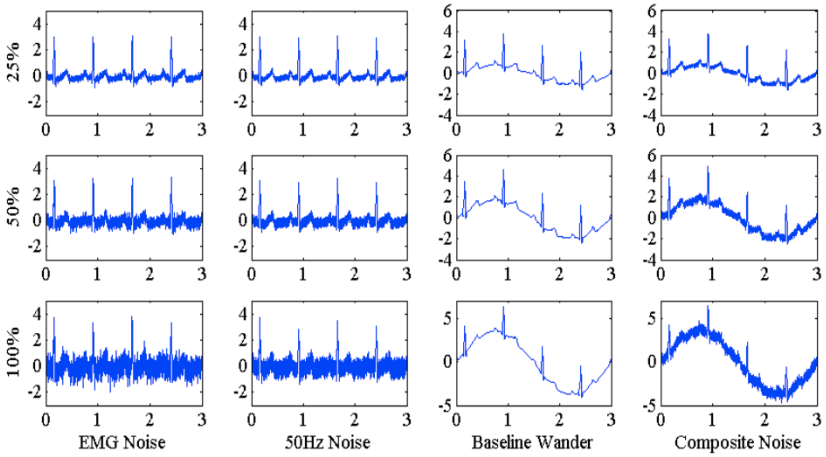
สัญญาณรบกวนในคลื่นไฟฟ้าหัวใจ คือ สัญญาณผิดปกติที่เกิดขึ้นในขณะบันทึกคลื่นไฟฟ้าหัวใจ ซึ่งอาจส่งผลต่อความแม่นยำของการตรวจจับสัญญาณการเต้นของหัวใจ โดยสัญญาณรบกวนที่พบได้บ่อยในคลื่นไฟฟ้าหัวใจและนำมาศึกษาในงานวิจัยนี้มี 4 แบบ (Chang, 2010) ได้แก่

1) Electromyographic Noise (EMG) เป็นสัญญาณรบกวนที่เกิดจากการหดเกร็งของกล้ามเนื้อหรือการเคลื่อนไหวของร่างกายอย่างกระทันหัน ความถี่ใน EMG ทับซ้อนกันมาก ทำให้ตรวจจับยาก

2) Powerline Interference (50 Hz) เป็นสัญญาณรบกวนความถี่สูง โดยทั่วไปจะมีความถี่ประมาณ 50 เฮิรตซ์ มีลักษณะเป็นสัญญาณรบกวนแบบคลื่นไซน์ อาจจะมาพร้อมฮาร์มอนิก (Harmonic) จำนวนหนึ่ง กล่าวคือ กระแสหรือแรงดันในรูปสัญญาณคลื่นไซน์ของสัญญาณหรือปริมาณในคาบใด ๆ ที่มีความถี่เป็นจำนวนเท่าของความถี่หลักมูล (Fundamental Frequency) สาเหตุหลักเกิดจากการรบกวนคลื่นแม่เหล็กไฟฟ้าจากสายไฟ (Electromagnetic Field : EMF) หรือจากเครื่องจักรในระยะใกล้ๆ การต่อสายดินที่ไม่ดี เป็นต้น

3) Baseline Wander เป็นสัญญาณรบกวนความถี่ต่ำมีลักษณะรบกวนจากแนวแกน x สูงขึ้นหรือลดต่ำลงจากแนวเส้นฐาน (Baseline) สาเหตุหลักเกิดจากการหายใจ การติดขั้วไฟฟ้าไม่ดี เป็นต้น

4) Composite Noise เป็นสัญญาณรบกวนที่เกิดจากการเกิดร่วมกันของสัญญาณรบกวนทั้ง 3 แบบที่กล่าวมา



ภาพที่ 9 ตัวอย่างสัญญาณรบกวนแต่ละแบบ ที่ระดับ 25% 50% และ 100%

หมายเหตุ: จาก “Arrhythmia ECG Noise Reduction by Ensemble Empirical Mode Decomposition” โดย Chang, K. M., 2010

บทที่ 3

วิธีการดำเนินงานวิจัย

งานวิจัยนี้จะใช้ข้อมูล ECG5000 ในประเภท 0 (จังหวะการเต้นของหัวใจปกติ) และประเภท 1 (ภาวะหัวใจห้องล่างเต้นผิดจังหวะแบบพีวีซี R-ON-T จะเหลือข้อมูลทั้งหมด 4,686 จังหวะการเต้นของหัวใจ แต่ละจังหวะมีความยาว 140 จุดเวลาและจำลองสัญญาณรบกวน 4 แบบ ได้แก่ 1) EMG Noise 2) Powerline Interference 3) Baseline Wander และ 4) Composite Noise แต่ละแบบปรับระดับของสัญญาณรบกวนไว้ที่ 25% 50% และ 100% จะได้ชุดข้อมูลที่ใช้ศึกษาทั้งหมด 13 ชุด ดังตารางที่ 6

ตารางที่ 6 ชุดข้อมูลที่ใช้ศึกษา

ชุดข้อมูล	EMG Noise	Powerline Interference (50Hz)	Baseline Wander	Composite Noise
1	-	-	-	-
2	25%	-	-	-
3	50%	-	-	-
4	100%	-	-	-
5	-	25%	-	-
6	-	50%	-	-
7	-	100%	-	-
8	-	-	25%	-
9	-	-	50%	-
10	-	-	100%	-
11	-	-	-	25%
12	-	-	-	50%
13	-	-	-	100%

3.1 การจำลองสัญญาณรบกวนในคลื่นไฟฟ้าหัวใจ

ในงานวิจัยนี้จะใช้วิธีการจำลองสัญญาณรบกวนในคลื่นไฟฟ้าหัวใจจาก “Arrhythmia ECG Noise Reduction by Ensemble Empirical Mode Decomposition” (Chang, 2010)

คำจำกัดความที่จะใช้ในการจำลองสัญญาณรบกวน

คำจำกัดความที่ 1 : Vpp คือ ระยะขจัดตั้งแต่จุดสูงสุดถึงจุดต่ำสุดของคลื่นไฟฟ้าหัวใจปกติ

คำจำกัดความที่ 2 : Reduced Ratio คือ อัตราส่วนลด

คำจำกัดความที่ 3 : Noise Levels คือ ระดับของสัญญาณรบกวน

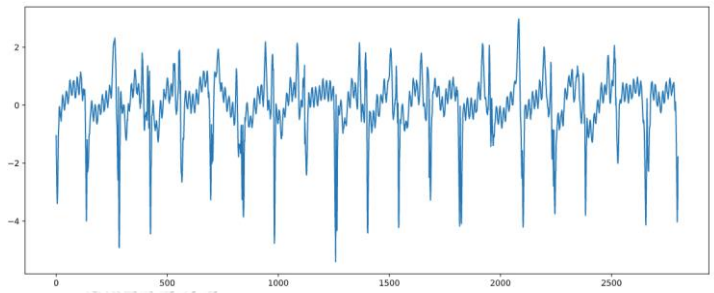
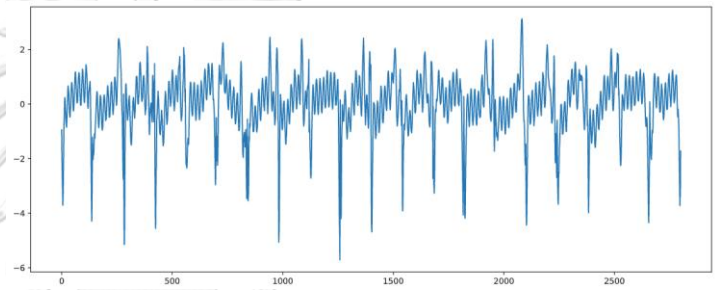
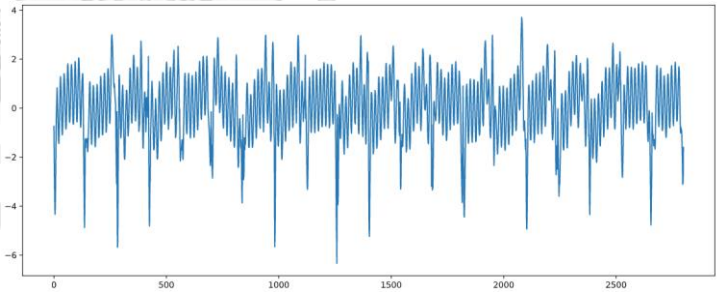
คำจำกัดความที่ 4 : Frequency คือ ความถี่ของสัญญาณรบกวน

3.1.1 Electromyography (EMG)

ตารางที่ 7 การจำลองสัญญาณรบกวน EMG

Noise Levels	Distribution	Maximum peak to minimum peak voltage: Vpp	Reduced Ratio
25% 50% 100%	Normal	5 mV	$\frac{1}{8}$

ตารางที่ 8 ตัวอย่างคลื่นไฟฟ้าหัวใจที่มีสัญญาณรบกวน EMG

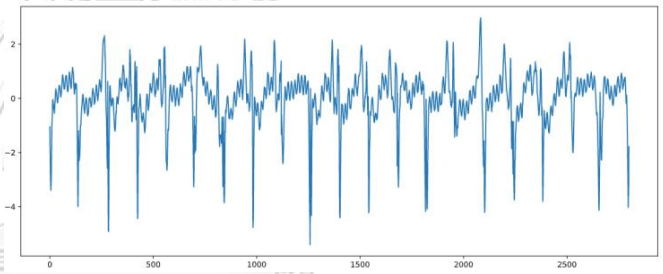
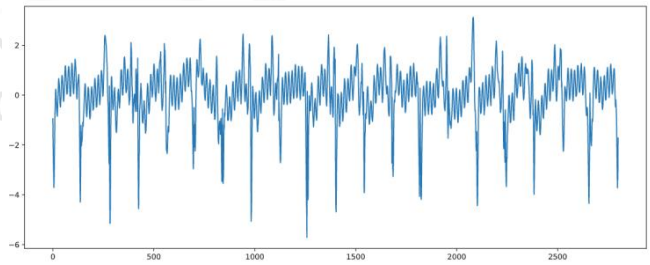
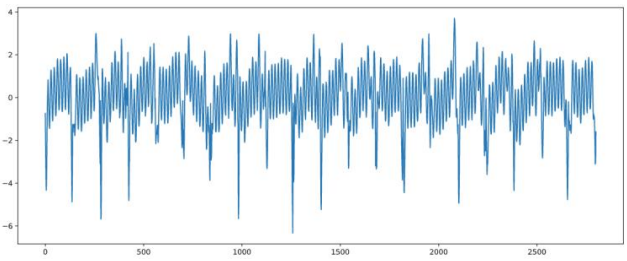
ระดับของสัญญาณรบกวน	ตัวอย่างคลื่นไฟฟ้าหัวใจ 20 จังหวะ ที่มีสัญญาณรบกวน EMG
25%	
50%	
100%	

3.1.2 Powerline Interference (50Hz)

ตารางที่ 9 การจำลองสัญญาณรบกวน Powerline Interference

Noise Levels	Function	Maximum peak to minimum peak voltage: Vpp	Frequency	Reduced Ratio
25% 50% 100%	Sine	5 mV	50 Hz	$\frac{1}{4}$

ตารางที่ 10 ตัวอย่างคลื่นไฟฟ้าหัวใจที่มีสัญญาณรบกวน Powerline Interference

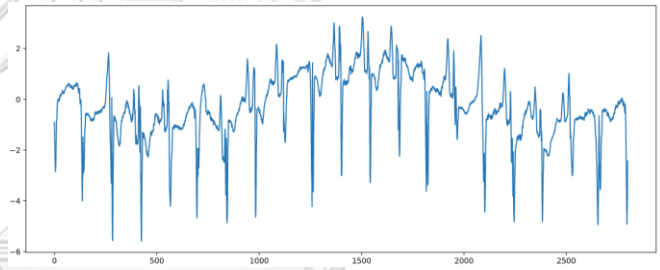
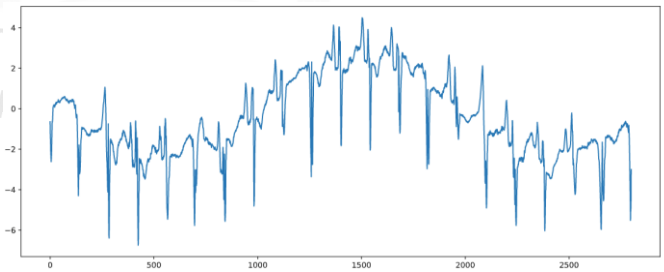
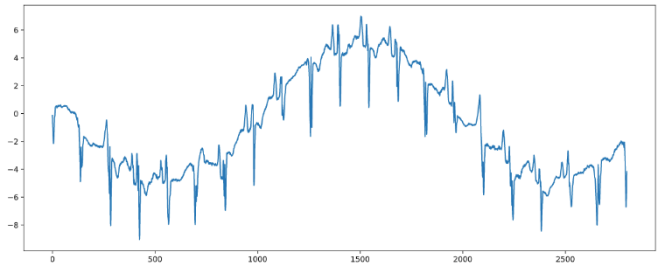
ระดับของสัญญาณรบกวน	ตัวอย่างคลื่นไฟฟ้าหัวใจ 20 จังหวะ ที่มีสัญญาณรบกวน Powerline Interference
25%	
50%	
100%	

3.1.3 Baseline Wander

ตารางที่ 11 การจำลองสัญญาณรบกวน Baseline Wander

Noise Levels	Function	Maximum peak to minimum peak voltage: Vpp	Frequency
25% 50% 100%	Sine	5 mV	0.33 Hz

ตารางที่ 12 ตัวอย่างคลื่นไฟฟ้าหัวใจที่มีสัญญาณรบกวน Baseline Wander

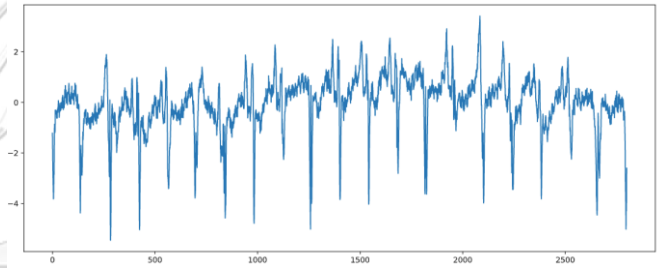
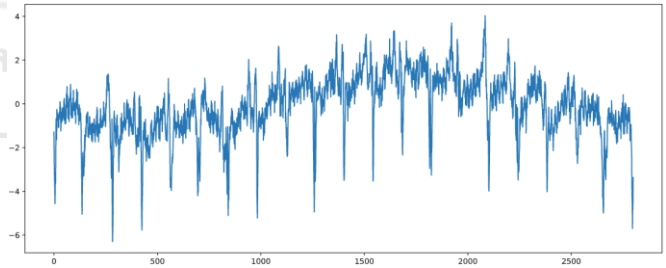
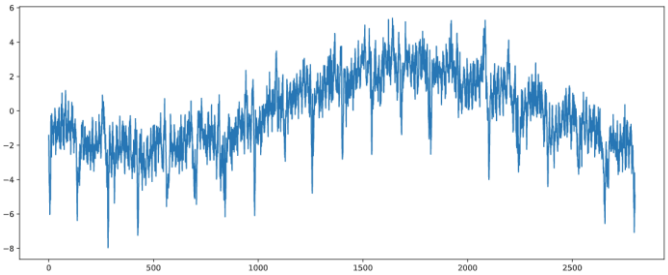
ระดับของสัญญาณรบกวน	ตัวอย่างคลื่นไฟฟ้าหัวใจ 20 จังหวะ ที่มีสัญญาณรบกวน Baseline Wander
25%	
50%	
100%	

3.1.4 Composite Noise

ตารางที่ 13 การจำลองสัญญาณรบกวน Composite

Noise Levels	Combination
25% 50% 100%	0.5 x (EMG Noise + Powerline Interference) + Baseline Noise

ตารางที่ 14 ตัวอย่างคลื่นไฟฟ้าหัวใจที่มีสัญญาณรบกวน Composite

ระดับของสัญญารบกวน	ตัวอย่างคลื่นไฟฟ้าหัวใจ 20 จังหวะ ที่มีสัญญาณรบกวน Composite
25%	
50%	
100%	

3.2 พารามิเตอร์ที่ใช้ใน SAX และ BOSS

ตารางที่ 15 พารามิเตอร์และความหมาย

พารามิเตอร์	ความหมาย
Window (w)	ขนาดหน้าต่างบานเลื่อน
Word (p)	ความยาวของตัวอักษร
Alphabet (a)	จำนวนตัวอักษร

3.3 วิธีดำเนินงาน

แสดงได้ดังภาพที่ 10

3.3.1 แบ่งข้อมูลออกเป็น 2 ส่วน คือ ชุดข้อมูลเรียนรู้ (Training Data) 70% และชุดทดสอบ (Testing Data) 30%

3.3.2 กำหนดขอบเขตล่างและขอบเขตบนสำหรับพารามิเตอร์แต่ละตัวและสุ่มเลือกชุดของพารามิเตอร์ (Window Size, Word Size, Alphabet Size) แต่ละชุด เพื่อหาค่าความถูกต้องโดยเฉลี่ยของพารามิเตอร์ชุดนั้น ๆ สำหรับ SAX และ BOSS โดยจะใช้ชุดข้อมูลเรียนรู้ (Training Data) ทั้งหมดมาทำ K-fold Cross Validation เพื่อแบ่งข้อมูลออกเป็นชุดข้อมูลเรียนรู้ (Training Data) และข้อมูลชุดตรวจสอบ (Validation Data) ในแต่ละส่วน (fold) มีขั้นตอนย่อย ดังนี้

ขั้นตอนที่ 1 ใช้ชุดข้อมูลเรียนรู้ (Training Data) หาความถี่ของรูปแบบตัวอักษร (Bag of patterns) โดยใช้วิธีของ SAX และ BOSS

ขั้นตอนที่ 2 สร้างเมทริกซ์ความถี่ของคำ (Term Frequency Matrix) ดังตารางที่ 16

ตารางที่ 16 ตัวอย่างเมทริซ์ความถี่ของคำ (Term Frequency Matrix)

ข้อมูลเรียนรู้ (Training Data)	aaccbabc	abcceaaa	bbcceaaa	babccaaa	...	ประเภท (Class)
1	2	10	3	1	...	1
2	1	3	2	11	...	0
:						:
n	...	...	...	...	...	...

ขั้นตอนที่ 3 สร้างเมทริกซ์น้ำหนักคำ (Term Weight Matrix) โดยคำนวณความถี่ของคำและการผกผันความถี่ในเอกสาร (tf-idf) ของรูปแบบต่าง ๆ ในแต่ละประเภท (Class) ดังตารางที่ 17

ตารางที่ 17 ตัวอย่าง Term Weight Matrix (TF-IDF)

Term	Class1	Class2	...	Class n
aabbcba	$\mathrm{tf} * \mathrm{idf}_{1,1}$	$\mathrm{tf} * \mathrm{idf}_{1,2}$	...	$\mathrm{tf} * \mathrm{idf}_{1,n}$
ccccbbaa	$\mathrm{tf} * \mathrm{idf}_{2,1}$	$\mathrm{tf} * \mathrm{idf}_{2,2}$	...	$\mathrm{tf} * \mathrm{idf}_{2,n}$
baccaacc	$\mathrm{tf} * \mathrm{idf}_{3,1}$	$\mathrm{tf} * \mathrm{idf}_{3,2}$	...	$\mathrm{tf} * \mathrm{idf}_{3,n}$
...	...	...	...	...

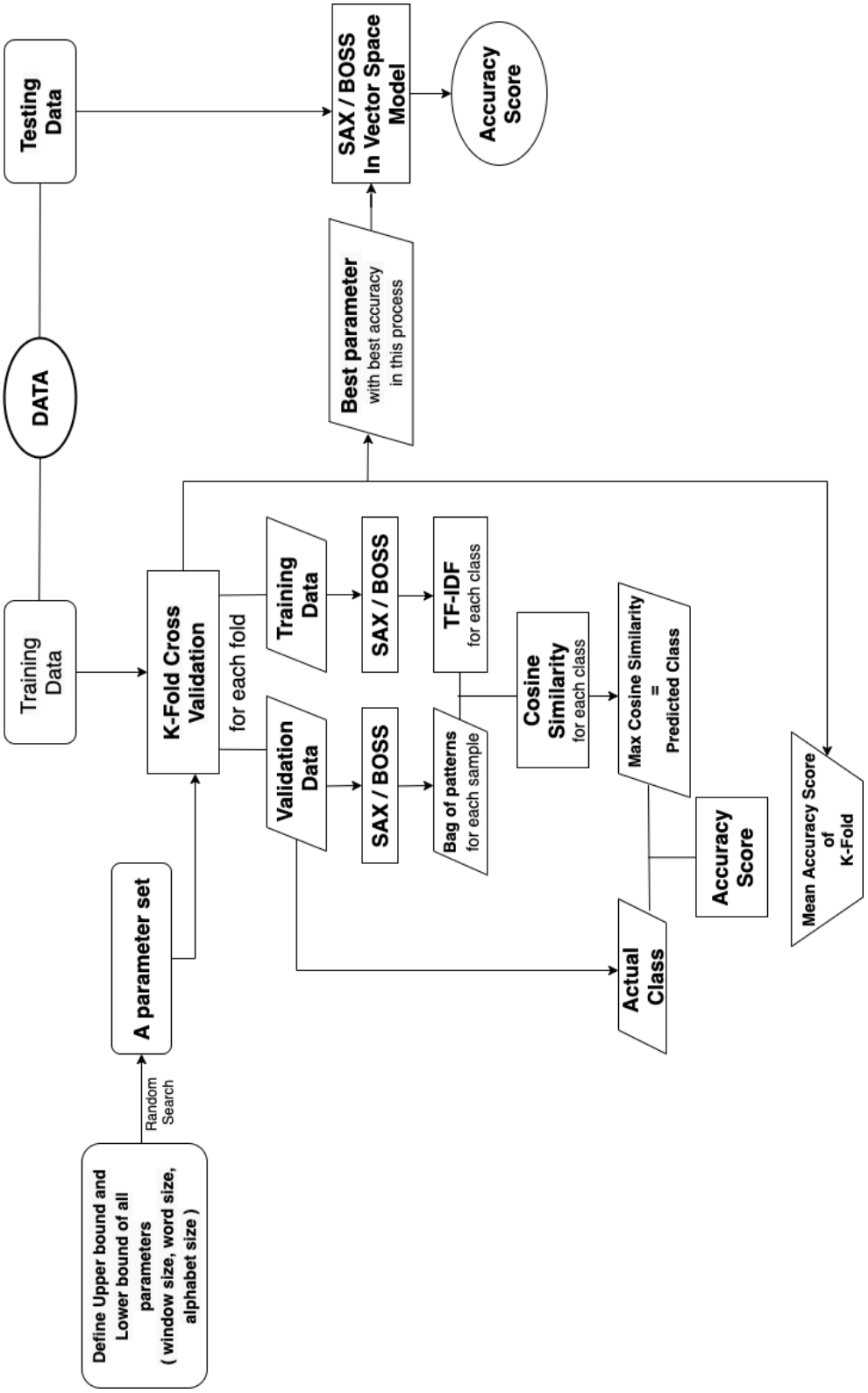
ขั้นตอนที่ 4 ใช้ชุดข้อมูลตรวจสอบ (Validation Data) หาความถี่ของรูปแบบตัวอักษร ตามขั้นตอนวิธีของ SAX และ BOSS

ขั้นตอนที่ 5 คำนวณค่าความเหมือนโคไซน์ (Cosine similarity) ระหว่าง tf-idf จากขั้นตอนที่ 3) กับความถี่ของตัวอักษรในชุดข้อมูลตรวจสอบ (Validation Data) จากขั้นตอนที่ 4) หาค่าความเหมือนโคไซน์ (Cosine Similarity) เพื่อจำแนกประเภทข้อมูล (Class) ให้กับชุดข้อมูลนี้ ถ้าประเภทใดมีค่าความเหมือนโคไซน์ (Cosine Similarity) สูงที่สุด จะทำนายว่าเป็นประเภทนั้น (Predicted Class)

ขั้นตอนที่ 6 คำนวณค่าความถูกต้อง (Accuracy) ระหว่างค่าจริง (Actual Class) กับค่าทำนาย (Predicted Class) ของชุดข้อมูลตรวจสอบ (Validation Data)

3.3.3 หาค่าเฉลี่ยของค่าความถูกต้องทั้ง K รอบสำหรับพารามิเตอร์ชุดนั้น ๆ และเลือกชุดของพารามิเตอร์ที่ดีที่สุดที่ให้ค่าเฉลี่ยค่าความถูกต้องสูงที่สุด

3.3.4 ใช้ชุดของพารามิเตอร์ที่ได้จากข้อ 3.3.3 ไปทดสอบกับชุดข้อมูลทดสอบ (Testing Data) และคำนวณค่าความถูกต้อง (Accuracy) คะแนน F1 (F1 Score) ค่าความแม่นยำ (Precision) และค่าความระลึก (Recall) สำหรับ SAX และ BOSS



ภาพที่ 10 วิธีดำเนินงาน

บทที่ 4

ผลการศึกษา

ในงานวิจัยนี้จะเขียนโปรแกรมโดยใช้ภาษา Python เวอร์ชั่น 3.6.13 ในการจำลองข้อมูล การวิเคราะห์ข้อมูลและการสร้างตัวแบบ และใช้ Tableau Desktop ในการทำ Visualization เพิ่มเติม และใช้บริการ Cloud Platform ในการสร้างเว็บแอพลิเคชันด้วย Heroku โดยจะใช้ข้อมูลทั้งหมด 13 ชุด ประกอบด้วยข้อมูลคลื่นไฟฟ้าหัวใจที่ไม่ได้เพิ่มสัญญาณรบกวนและข้อมูลที่มีการเพิ่มสัญญาณรบกวน 4 แบบ ได้แก่ 1) EMG Noise 2) Powerline Interference 3) Baseline Wander และ 4) Composite Noise แต่ละแบบปรับระดับของสัญญาณรบกวนไว้ที่ 25% 50% และ 100% สำหรับข้อมูลแต่ละชุดจะใช้ตัวแปรและเขียนคำอธิบายไว้ดังตารางที่ 18 และตัวอย่างของข้อมูลคลื่นไฟฟ้าหัวใจที่นำมาใช้ศึกษาจะเป็นข้อมูลอนุกรมเวลา 4,686 ตัวอย่าง ความยาว 140 และมีประเภทกำกับ (Class) ดังตารางที่ 19

4.1 คำอธิบายและตัวอย่างของข้อมูล

ตารางที่ 18 คำอธิบายข้อมูล

ข้อมูล	คำอธิบาย
ECG	ข้อมูลคลื่นไฟฟ้าหัวใจ
data_emg_noise25	ข้อมูลคลื่นไฟฟ้าหัวใจ (ECG) ที่มีสัญญาณรบกวนประเภท EMG ที่ระดับ 25%
data_emg_noise50	ข้อมูลคลื่นไฟฟ้าหัวใจ (ECG) ที่มีสัญญาณรบกวนประเภท EMG ที่ระดับ 50%
data_emg_noise100	ข้อมูลคลื่นไฟฟ้าหัวใจ (ECG) ที่มีสัญญาณรบกวนประเภท EMG ที่ระดับ 100%
data_power_noise25	ข้อมูลคลื่นไฟฟ้าหัวใจ (ECG) ที่มีสัญญาณรบกวนประเภท Powerline ที่ระดับ 25%
data_power_noise50	ข้อมูลคลื่นไฟฟ้าหัวใจ (ECG) ที่มีสัญญาณรบกวนประเภท Powerline ที่ระดับ 50%
data_power_noise100	ข้อมูลคลื่นไฟฟ้าหัวใจ (ECG) ที่มีสัญญาณรบกวนประเภท Powerline ที่ระดับ 100%

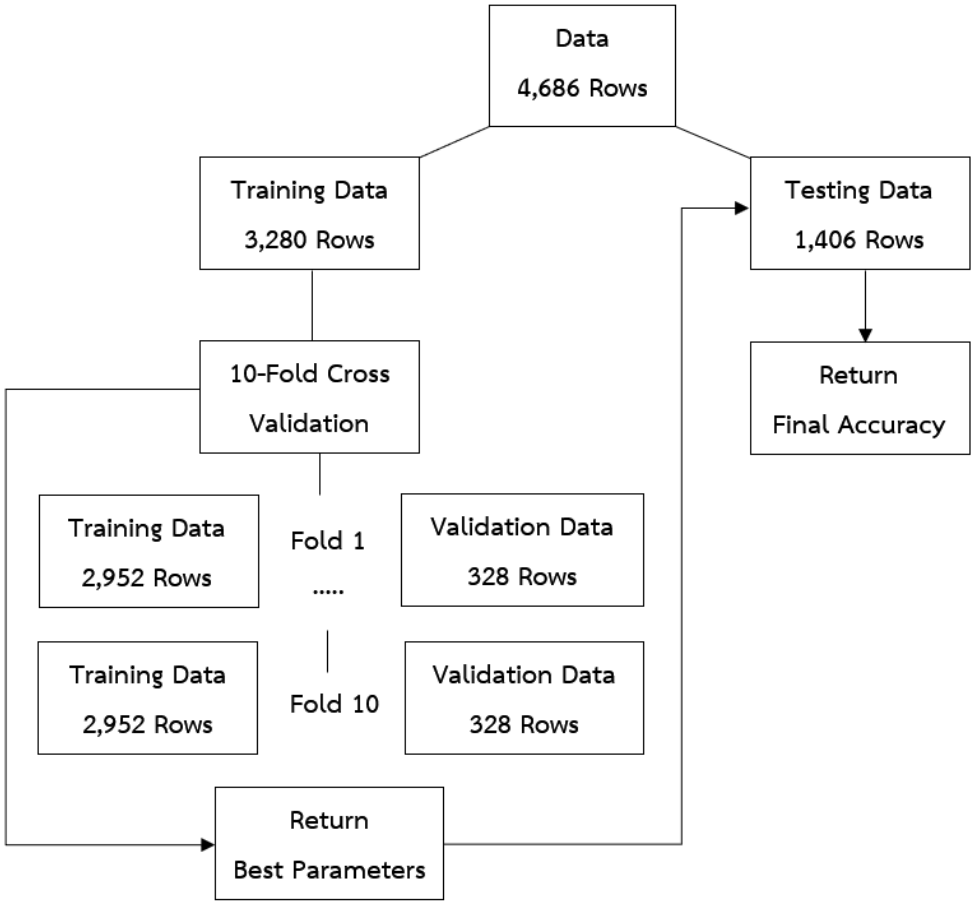
data_base_noise25	ข้อมูลคลื่นไฟฟ้าหัวใจ (ECG) ที่มีสัญญาณรบกวนประเภท Baseline ที่ระดับ 25%
data_base_noise50	ข้อมูลคลื่นไฟฟ้าหัวใจ (ECG) ที่มีสัญญาณรบกวนประเภท Baseline ที่ระดับ 50%
data_base_noise100	ข้อมูลคลื่นไฟฟ้าหัวใจ (ECG) ที่มีสัญญาณรบกวนประเภท Baseline ที่ระดับ 100%
data_comb_noise25	ข้อมูลคลื่นไฟฟ้าหัวใจ (ECG) ที่มีสัญญาณรบกวนประเภท Composite ที่ระดับ 25%
data_comb_noise50	ข้อมูลคลื่นไฟฟ้าหัวใจ (ECG) ที่มีสัญญาณรบกวนประเภท Composite ที่ระดับ 50%
data_comb_noise100	ข้อมูลคลื่นไฟฟ้าหัวใจ (ECG) ที่มีสัญญาณรบกวนประเภท Composite ที่ระดับ 100%
Class	ประเภทของข้อมูลคลื่นไฟฟ้าหัวใจ 0 : จังหวะการเต้นของหัวใจปกติ 1 : จังหวะการเต้นของหัวใจห้องล่างเต้นผิดจังหวะแบบพีวีซี R-ON-T
$t_i = (t_1, t_2, \ldots, t_{140})$	ข้อมูลคลื่นไฟฟ้าหัวใจตามลำดับเวลา

ตารางที่ 19 ตัวอย่างของข้อมูลคลื่นไฟฟ้าหัวใจ

No.	t_1	t_2	t_3	...	t_{140}	Class
1	-0.6566696	-1.1872677	-1.2823869	...	0.2257076	1
2	0.3973526	-1.6893153	-2.7698923	...	0.52316457	0
3	-1.862747	-3.2036327	-3.6341531	...	-0.6100227	0
4	-0.9572602	-2.1521446	-2.7491046	...	-1.727039	1
5	0.91324841	-0.1049549	-0.9147156	...	-0.4820698	1
...	...	...	...	...	...	...
...	...	...	...	...	...	...
...	...	...	...	...	...	...
4686	-1.7946848	-3.0047093	-3.2965459	...	-1.2108684	0

4.2 การแบ่งข้อมูล

จากข้อมูลทั้งหมดจะถูกแบ่งข้อมูลออกเป็น 2 ส่วน คือ ข้อมูลชุดเรียนรู้ (Training Data) สัดส่วน 70% (3,280 ตัวอย่าง) และข้อมูลชุดทดสอบ (Testing Data) สัดส่วน 30% (1,406 ตัวอย่าง) สำหรับข้อมูลชุดเรียนรู้ (Training Data) สัดส่วน 70% จะนำไปทำ 10-Fold Cross Validation เพื่อหาพารามิเตอร์ที่ดีที่สุด จะได้ว่าในแต่ละส่วน (Fold) จะมีข้อมูลชุดเรียนรู้ (Training Data) 2,952 ตัวอย่าง และข้อมูลชุดตรวจสอบ (Validation Data) 328 ตัวอย่าง ส่วนข้อมูลชุดทดสอบ (Testing Data) สัดส่วน 30% จะนำไปทดสอบความถูกต้อง ดังภาพที่ 11



ภาพที่ 11 แผนภาพการแบ่งข้อมูล

4.3 ผลการวิเคราะห์ข้อมูล

จากข้อมูลชุดเรียนรู้ (Training Data) 3,280 ตัวอย่าง นำมาใช้ทำ 10 Fold Cross Validation จะได้ว่าในแต่ละส่วน (Fold) จะเหลือข้อมูลชุดเรียนรู้ (Training Data) 2,952 ตัวอย่าง และข้อมูลชุดตรวจสอบ (Validation Data) 328 ตัวอย่าง ในข้อมูลชุดเรียนรู้ 2,952 ตัวอย่างจะนำมาใช้สร้างเมริกซ์น้ำหนักคำ (Term Weight Matrix) สำหรับแต่ละส่วน (Fold) และสำหรับข้อมูลชุดตรวจสอบ (Validation Data) จะใช้เปรียบเทียบค่าความเหมือนโคไซน์ (Cosine Similarity) ระหว่างข้อมูลแต่ละประเภท (Class) สามารถแสดงได้ตัวอย่างตารางต่อไปนี้

หมายเหตุ เนื่องจากมีข้อมูลหลายชุดและมีการวิเคราะห์ข้อมูลหลายขั้นตอน ทำให้ได้ข้อมูลปริมาณมาก ดังนั้นในส่วนของการวิเคราะห์ข้อมูลจึงเป็นเพียงตารางตัวอย่างของข้อมูลในหัวข้อนั้น ๆ ทั้งนี้และผู้วิจัยได้แนบ Code ไว้ในภาคผนวก และสามารถดูผลการวิเคราะห์ข้อมูลเพิ่มเติมได้ที่ https://ecg-project-napatsorn.herokuapp.com

4.3.1 ตัวอย่างเมทริกซ์น้ำหนักคำ (Term Weight Matrix) ในแต่ละส่วน (Fold)

ในขั้นตอนการทำ K-Fold Cross Validation ข้อมูลในแต่ละส่วน (Fold) จะถูกแบ่งเป็นข้อมูลชุดเรียนรู้ (Training Data) 2,952 ตัวอย่าง และข้อมูลชุดตรวจสอบ (Validation Data) 328 ตัวอย่าง ในแต่ละกลุ่ม (Fold) จะใช้ข้อมูลชุดเรียนรู้ (Training Data) เพื่อสร้างเมทริกซ์น้ำหนักคำ (Term Weight Matrix) ด้วย Term Frequency-Inverse Document Frequency (TF-IDF) สำหรับข้อมูลในแต่ละประเภท (Class) เพื่อนำค่านี้ไปคำนวณค่าความเหมือน (Cosine Similarity) กับข้อมูลในข้อมูลชุดตรวจสอบ (Validation Data)

ตารางที่ 20 ตัวอย่างเมทริกซ์น้ำหนักคำ (Term Weight Matrix) สำหรับข้อมูลแต่ละส่วน (Fold)

Fold	Class	aacacacccc	aacacacacc	aacacacaca	aabacacaca	...
1	0	0	0	1	1	...
	1	10.2648657	10.1541631	6.70044357	1	...
2	0	0	0	1	1	...
	1	10.2324082	10.0959736	6.63121178	1	...
3	0	0	0	1	1	...
	1	10.2214493	10.210419	6.69373214	1	...
4	0	0	0	0	1	...
	1	10.1768902	10.1655648	11.3448371	1	...
5	0	0	0	1	1	...
	1	10.2324082	10.210419	6.69373214	1	...

6	0	0	0	1	1	...
	1	10.2324082	10.210419	6.68357977	1	...
7	0	0	0	1	0	...
	1	10.2541155	10.072126	6.68697536	1.69314718	...
8	1	1.69314718	1.69314718	0	0	...
	0	0	0	1.69314718	1.69314718	...
9	1	10.2648657	10.1768902	6.7235851	0	...
	0	0	0	1	1.69314718	...
10	0	0	0	1	1	...
	1	10.3071966	10.2324082	6.72031178	1	...

4.3.2 ตัวอย่างเมทริซ์ความถี่ของคำ (Term Frequency Matrix)

ในขั้นตอนการทำ K-Fold Cross Validation ข้อมูลในแต่ละส่วน (Fold) จะถูกแบ่งเป็นข้อมูลชุดเรียนรู้ (Training Data) 2,952 ตัวอย่าง และข้อมูลชุดตรวจสอบ (Validation Data) 328 ตัวอย่าง ในแต่ละกลุ่ม (Fold) จะใช้ข้อมูลชุดตรวจสอบ (Validation Data) สร้างเมทริกซ์ความถี่ของคำ (Term Frequency Matrix)

ตารางที่ 21 ตัวอย่างเมทริซ์ความถี่ของคำ (Term Frequency Matrix)

Fold	ชุดข้อมูลตรวจสอบ (Validation Data)	aacacacccc	aacacacacc	aacacacaca	aabacacaca	...
1	1	1	2	10	5	...
1	2	10	2	2	1	...
1	...	...	...	...	...	...
1	328	2	4	1	4	...
2	1	1	1	2	14	...
2	2	0	1	15	2	...
2	...	...	...	...	...	...
2	328	0	3	4	1	...
10	1	1	3	2	1	...
10	2	3	0	7	9	...
10	....	...	...	...	...	...
10	328	6	10	0	0	...

4.3.3 ตัวอย่างเมทริกซ์ค่าความเหมือนโคไซน์ (Cosine Similarity)

จากเมทริกซ์น้ำหนักคำ (TF-IDF) ในตารางที่ 20 และเมทริกซ์ความถี่ของคำ (Term Frequency Matrix) ในตารางที่ 21 สามารถนำมาคำนวณค่าความเหมือนโคไซน์ (Cosine Similarity) ในข้อมูลแต่ละประเภท (Class)

ตารางที่ 22 ตัวอย่างค่าความเหมือนโคไซน์ (Cosine Similarity) สำหรับข้อมูลแต่ละส่วน (Fold)

Fold	ชุดข้อมูลตรวจสอบ (Validation Data)	Cosine Similarity		ค่าที่ทำนาย (Predicted Class)
		Class 0	Class 1	
1	1	0.18618717	0.02491039	0
1	2	0.0330811	0.13521933	1
1	3	0.0747216	0.05583699	0
1	...	...	...	...
1	328	...	...	1
2	1	0.20530389	0.03288683	1
2	2	0.19829105	0.03587269	1
2	3	0.18662884	0.03801201	0
2	...	...	...	...
2	328	...	...	0
...	...	...	...	...
...	...	...	...	...
10	1	0.03568761	0.21033327	0
10	2	0.08044959	0.06238019	1
10	3	0.04485259	0.17304159	1
10	...	...	...	...
10	328	...	...	0

4.3.4 ตัวอย่างผลการเปรียบเทียบค่าความถูกต้องระหว่างค่าจริง (Actual Class) และค่าทำนาย (Predicted Class)

จากการคำนวณค่าความเหมือนโคไซน์ (Cosine Similarity) จะได้ค่าทำนาย (Predicted Class) และจากข้อมูลชุดตรวจสอบ (Validation) จะได้ค่าจริง (Actual Class) นำมาเปรียบเทียบและคำนวณค่าความถูกต้อง (Accuracy)

ตารางที่ 23 ตัวอย่างผลการเปรียบเทียบค่าความถูกต้องระหว่างค่าจริง (Actual Class) และค่าทำนาย (Predicted Class)

Fold	ชุดข้อมูลตรวจสอบ (Validation Data)	ค่าจริง (Actual Class)	ค่าที่ทำนาย (Predicted Class)	ผลการทำนาย	ค่าความถูกต้อง (Accuracy)
1	1	1	0	False	0.80
1	2	1	1	True	
1	3	0	0	True	
1	...	...	...	...	
1	328	0	1	False	
2	1	0	1	False	0.76
2	2	1	1	True	
2	3	1	0	True	
2	...	...	...		
2	328	1	0	False	
...	...	...	...	...	...
...	...	...	...	...	
...	...	...	...	...	
10	1	1	0	False	0.92
10	2	1	1	True	
10	3	1	1	True	
10	...	...	...	...	
10	328	0	0	True	
ค่าเฉลี่ยของค่าความถูกต้องสำหรับพารามิเตอร์ชุดใด ๆ					0.83

4.3.5 ตัวอย่างสรุปค่าความถูกต้องเฉลี่ยสำหรับพารามิเตอร์และข้อมูลแต่ละชุด

ในแต่ละชุดของพารามิเตอร์ นำค่าความถูกต้อง (Accuracy) ที่ได้จากทุก ๆ ส่วน (Fold) มาคำนวณเป็นค่าค่าความถูกต้องโดยเฉลี่ย (Mean Accuracy) สำหรับพารามิเตอร์แต่ละชุด

ตารางที่ 24 ตัวอย่างผลสรุปค่าความถูกต้องโดยเฉลี่ยสำหรับพารามิเตอร์แต่ละชุด

Parameter No.	Data	Parameters			Mean Accuracy	
		Window	Word	Alphabet	SAXVSM	BOSSVS
1	ECG	12	10	9	0.994432	0.993875
2	ECG	37	9	8	0.923244	0.963456
...	ECG	...	...	...		
30	ECG	46	31	3	0.989042	0.923502
1	data_emg_noise25	12	10	9	0.953562	0.952455
2	data_emg_noise25	37	9	8	0.945333	0.912442
...	data_emg_noise25	...	...	...		
30	data_emg_noise25	46	31	3	0.905425	0.943254
...	...	...	...	...	...	...
...	...	...	...	...	...	...
1	data_comb_noise100	12	10	9	0.693451	0.773456
2	data_comb_noise100	37	9	8	0.664524	0.604143
...	data_comb_noise100	...	...	...		
30	data_comb_noise100	46	31	3	0.714452	0.790425

4.3.6 สรุปพารามิเตอร์ที่ดีที่สุดสำหรับข้อมูลชุดเรียนรู้ (Training Data) ที่จะนำไปใช้ทดสอบกับข้อมูลชุดทดสอบ (Testing Data)

จากค่าความถูกต้องโดยเฉลี่ย (Mean Accuracy) ในตารางที่ 24 นำมาเปรียบเทียบกันเพื่อเลือกพารามิเตอร์ที่ดีที่สุดที่ให้ค่าความถูกต้องสูงที่สุดสำหรับข้อมูลในแต่ละชุด นำมาใช้ทดสอบกับข้อมูลชุดทดสอบ (Testing Data)

ตารางที่ 25 สรุปพารามิเตอร์ที่ดีที่สุดสำหรับข้อมูลชุดเรียนรู้ (Training Data)

Data	Model	Window	Word	Alphabet	Best Accuracy	SD
ECG	SAX	44	10	5	0.9945122	0.00355546
	BOSS	48	11	6	0.99146341	0.00328364
Data_emg_noise25	SAX	44	10	5	0.99664634	0.00287621
	BOSS	41	11	5	0.99420732	0.00213415
Data_emg_noise50	SAX	44	10	5	0.99634146	0.00355546
	BOSS	40	7	3	0.9945122	0.00448077
Data_emg_noise100	SAX	48	11	6	0.99329268	0.00328364
	BOSS	40	7	3	0.99329268	0.00404466
Data_power_noise25	SAX	37	9	8	0.99329268	0.00328364
	BOSS	48	11	6	0.98993902	0.00362023
Data_power_noise50	SAX	48	11	6	0.9902439	0.00487805
	BOSS	48	11	6	0.98871951	0.00453234
Data_power_noise100	SAX	45	12	8	0.9804878	0.00435453
	BOSS	45	12	8	0.98871951	0.00640244
Data_base_noise25	SAX	45	12	8	0.98323171	0.00598214
	BOSS	44	10	5	0.98993902	0.00546234
Data_base_noise50	SAX	41	11	5	0.98140244	0.00616578
	BOSS	44	10	5	0.99054878	0.00419138
Data_base_noise100	SAX	48	11	6	0.99329268	0.00506501
	BOSS	27	14	2	0.98871951	0.00625557
Data_comb_noise25	SAX	44	10	5	0.99512195	0.00279425
	BOSS	44	10	5	0.99481707	0.00335366
Data_comb_noise50	SAX	44	10	5	0.99542683	0.00391623
	BOSS	44	10	5	0.99481707	0.00306399
Data_comb_noise100	SAX	44	10	5	0.99146341	0.00265787
	BOSS	44	10	5	0.99268293	0.00310916

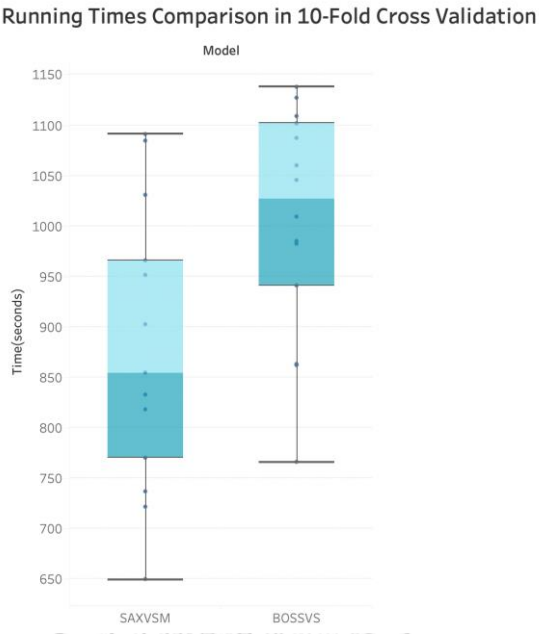
4.3.7 การเปรียบเทียบประสิทธิภาพของ SAXVSM และ BOSSVS ในข้อมูลทดสอบ (Testing Data)

ใช้พารามิเตอร์ที่ดีที่สุดจากการทำ K-Fold Cross Validation ในข้อมูลชุดเรียนรู้ (Training Data) มาใช้ทดสอบกับข้อมูลทดสอบ (Testing Data) เพื่อเปรียบเทียบค่าความถูกต้อง (Accuracy) คะแนน F1 (F1 Score) ค่าความแม่นยำ (Precision) ค่าความระลึก (Recall) ของแต่ละตัวแบบ

ตารางที่ 26 เปรียบเทียบประสิทธิภาพของ SAXVSM และ BOSSVS ในข้อมูลทดสอบ

data	Model	Accuracy	F1 Score	Precision	Recall
ECG	SAX	0.99075391	0.98751201	0.98467433	0.99036609
	BOSS	0.99075391	0.98748797	0.98653846	0.98843931
data_emg_noise25	SAX	0.9943101	0.99227799	0.99419729	0.99036609
	BOSS	0.99004267	0.98653846	0.98464491	0.98843931
data_emg_noise50	SAX	0.99146515	0.98846154	0.9865643	0.99036609
	BOSS	0.99359886	0.99132112	0.99227799	0.99036609
data_emg_noise100	SAX	0.98933144	0.98561841	0.98091603	0.99036609
	BOSS	0.99217639	0.9894129	0.98846154	0.99036609
data_power_noise25	SAX	0.98577525	0.98091603	0.97164461	0.99036609
	BOSS	0.99146515	0.98846154	0.9865643	0.99036609
data_power_noise50	SAX	0.98933144	0.98570067	0.9754717	0.99614644
	BOSS	0.9886202	0.98470363	0.9772296	0.99229287
data_power_noise100	SAX	0.97937411	0.97261568	0.9537037	0.99229287
	BOSS	0.98933144	0.9855352	0.98648649	0.98458574
data_base_noise25	SAX	0.98435277	0.97920605	0.96103896	0.99807322
	BOSS	0.99075391	0.98758357	0.97916667	0.99614644
data_base_noise50	SAX	0.98790896	0.98388626	0.96828358	1
	BOSS	0.99288762	0.99045802	0.98109641	1
data_base_noise100	SAX	0.99288762	0.99043977	0.9829222	0.99807322
	BOSS	0.98008535	0.97297297	0.97485493	0.97109827
data_comb_noise25	SAX	0.99004267	0.98653846	0.98464491	0.98843931
	BOSS	0.99146515	0.98846154	0.9865643	0.99036609
data_comb_noise50	SAX	0.99288762	0.99040307	0.98661568	0.99421965
	BOSS	0.99004267	0.98653846	0.98464491	0.98843931
data_comb_noise100	SAX	0.99217639	0.98949379	0.98106061	0.99807322
	BOSS	0.98719772	0.98272553	0.9789675	0.98651252

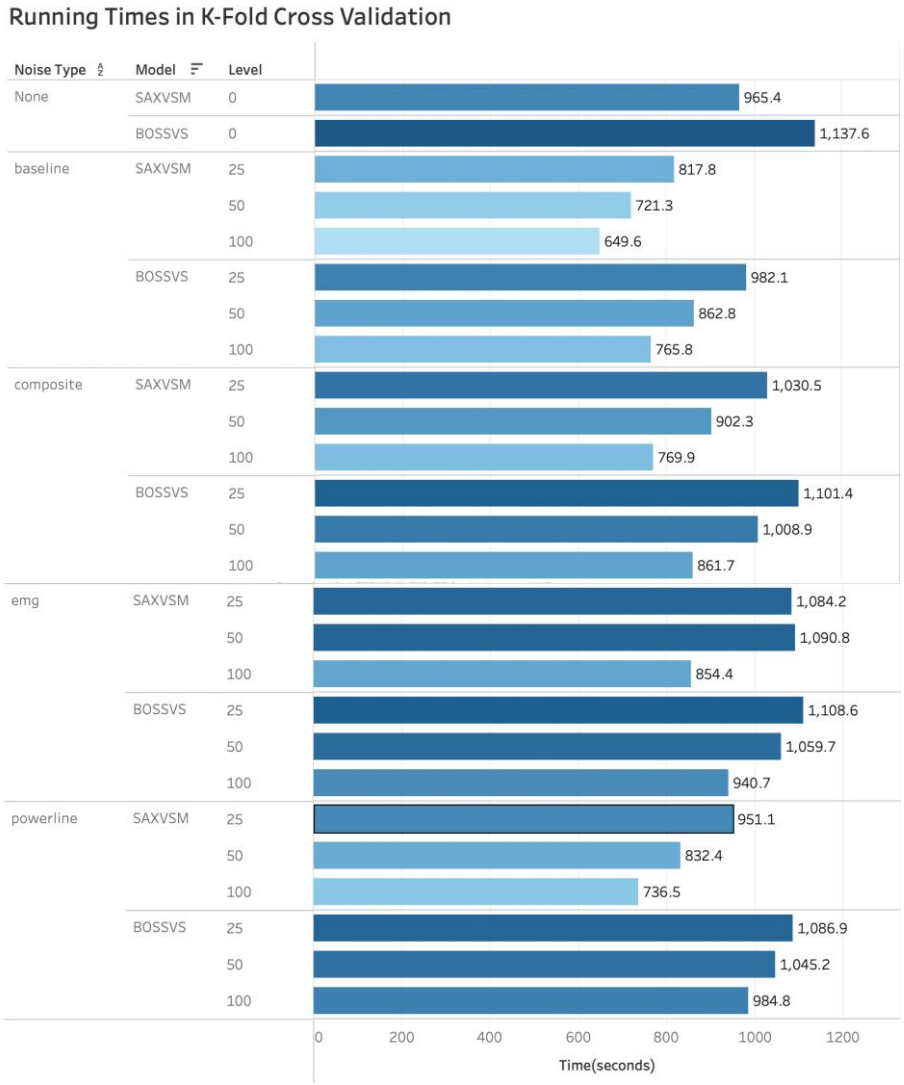
4.3.8 การเปรียบเทียบเวลาที่ใช้ในขั้นตอนการสอนตัวแบบด้วย 10-Fold Cross Validation ระหว่าง SAXVSM และ BOSSVS



ภาพที่ 12 Boxplot เปรียบเทียบเวลาที่ใช้ในขั้นตอนการสอนตัวแบบระหว่าง SAXVSM และ BOSSVS สำหรับข้อมูลคลื่นไฟฟ้าหัวใจที่มีสัญญาณรบกวนทั้ง 4 แบบ ที่ระดับ 25% 50% และ 100%

จาก Box plot แสดงเวลาที่ใช้ในขั้นตอนการสอนตัวแบบด้วย 10-Fold Cross Validation เมื่อทดสอบกับ SAXVSM และ BOSSVS เพื่อหาพารามิเตอร์ที่ดีที่สุดสำหรับข้อมูลแต่ละชุด จะเห็นว่า BOSSVS ใช้เวลาประมาณ 950-1,100 วินาที หรือ 15-18 นาที ต่อข้อมูล 1 ชุด ในขณะที่ SAXVSM ใช้เวลาประมาณ 650-950 วินาที หรือ 10-16 นาที ต่อข้อมูล 1 ชุด

4.3.9 การเปรียบเทียบเวลาที่ใช้ในขั้นตอนการสอนตัวแบบด้วย K-Fold Cross Validation แบ่งตามข้อมูลที่มีสัญญาณรบกวนแต่ละแบบ ในระดับต่าง ๆ

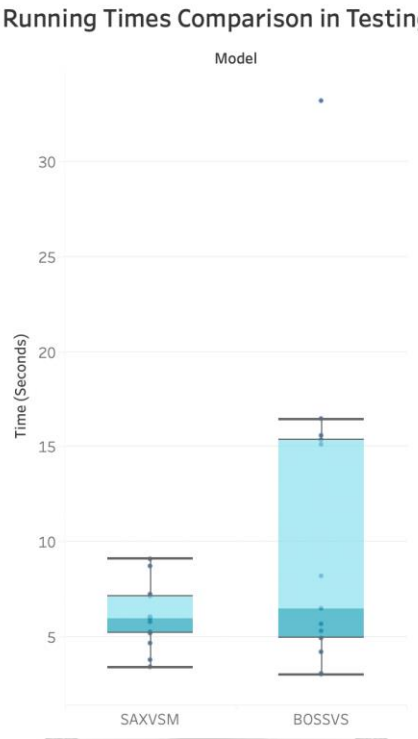


ภาพที่ 13 กราฟแท่งเปรียบเทียบเวลาที่ใช้ในการทดสอบตัวแบบ
สำหรับข้อมูลคลื่นไฟฟ้าหัวใจที่มีสัญญาณรบกวนทั้ง 4 แบบ ที่ระดับ 25% 50% และ 100%

จากกราฟแท่ง แสดงเวลาที่ใช้ในขั้นตอนการสอนตัวแบบด้วย 10-Fold Cross Validation ในข้อมูลคลื่นไฟฟ้าหัวใจที่มีสัญญาณรบกวนแบบต่าง ๆ ที่ระดับต่าง ๆ จะเห็นว่ายิ่งมีการเพิ่มสัญญาณรบกวนในระดับที่สูงขึ้นทั้ง SAXVSM และ BOSSVS จะใช้เวลาในการสอนตัวแบบน้อยลง เนื่องจากในขั้นตอนการสร้างเมทริกซ์น้ำหนักของคำด้วยการหาความถี่ของคำและการผกผันความถี่ในเอกสาร (TF-IDF Matrix) ยิ่งข้อมูลที่มีระดับสัญญาณรบกวนสูงขึ้น ทำให้ตัวแบบเจอรูปแบบของสัญญาณ

รบกวนมาบดบังรูปแบบที่แท้จริงของคลื่นไฟฟ้าหัวใจ ทำให้ได้จำนวนรูปแบบตัวอักษรระหว่างประเภทที่ปกติและผิดปกติซ้ำกันมากขึ้น และความหลากหลายของรูปแบบตัวอักษรน้อยลง จึงทำให้เมทริกซ์มีขนาดเล็กลง เมื่อไปคำนวณค่าความเหมือนโคไซน์ (Cosine Similarity) จึงทำได้รวดเร็วขึ้น

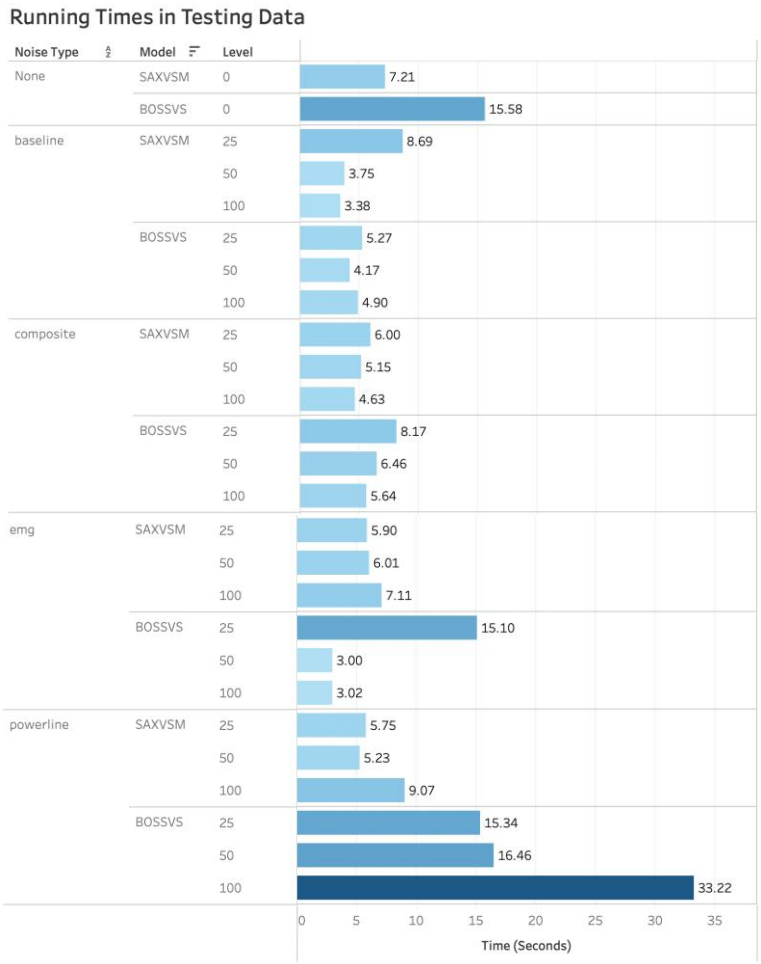
4.3.10 การเปรียบเทียบเวลาที่ใช้ในการทดสอบตัวแบบระหว่าง SAXVSM และ BOSSVS



ภาพที่ 14 Box plot เปรียบเทียบเวลาในการทดสอบตัวแบบระหว่าง SAXVSM และ BOSSVS สำหรับข้อมูลคลื่นไฟฟ้าหัวใจที่มีสัญญาณรบกวนทั้ง 4 แบบ ที่ระดับ 25% 50% และ 100%

จาก Box plot แสดงเวลาที่ใช้ในการทดสอบตัวแบบสำหรับ Testing Data ระหว่าง SAXVSM และ BOSSVS จะเห็นว่า BOSSVS ใช้เวลานานกว่า SAXVSM และมีการกระจายตัวมากกว่า อยู่ที่ประมาณ 5-15 วินาที ต่อข้อมูล 1 ชุด ส่วน SAXVSM ใช้เวลาอยู่ที่ประมาณ 5-8 วินาที ต่อข้อมูล 1 ชุด

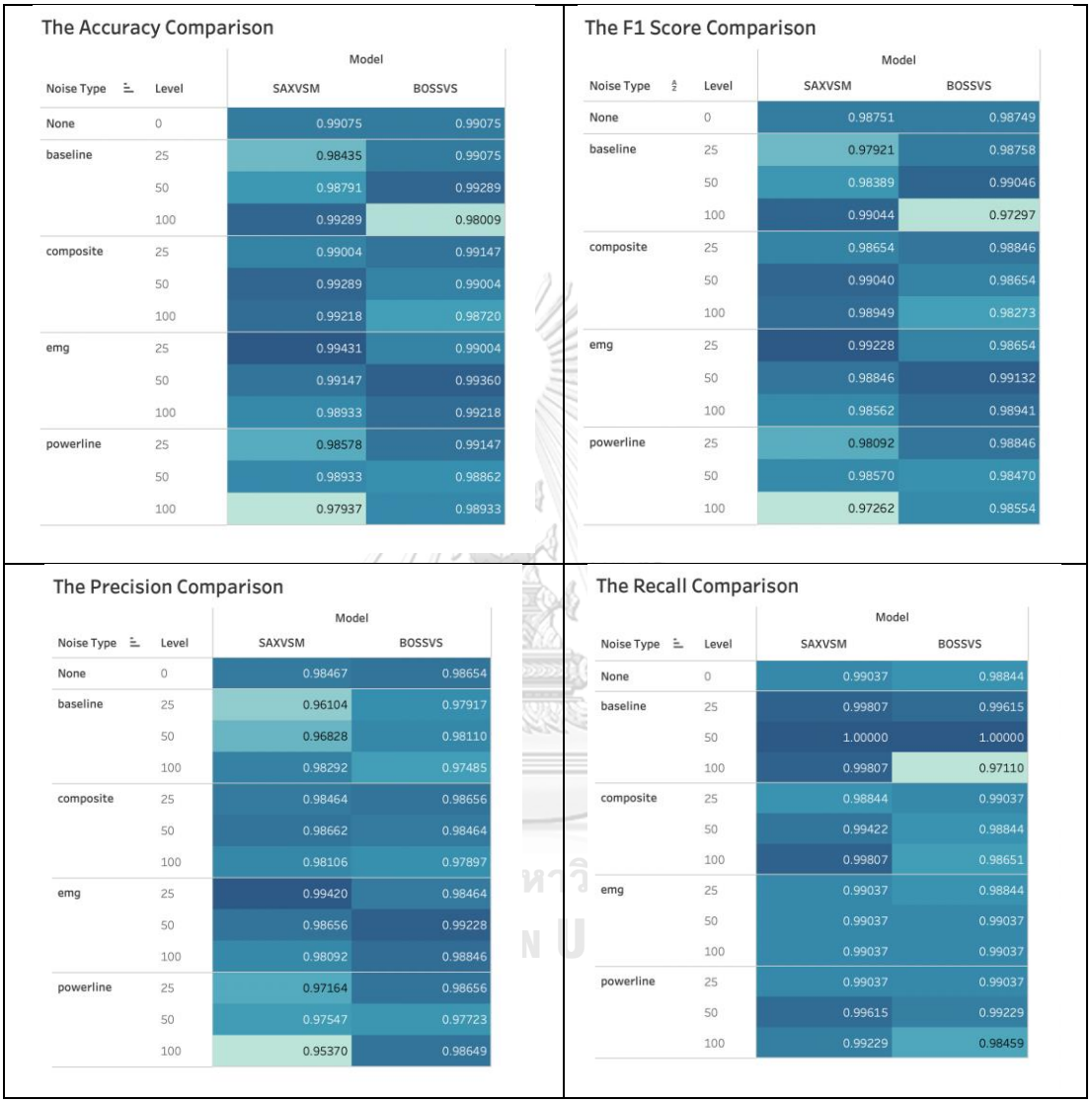
4.3.11 การเปรียบเทียบเวลาที่ใช้ในการทดสอบตัวแบบ แบ่งตามสัญญาณรบกวนระดับต่าง ๆ



ภาพที่ 15 กราฟแท่งเปรียบเทียบเวลาที่ใช้ในการทดสอบตัวแบบ
สำหรับข้อมูลคลื่นไฟฟ้าหัวใจที่มีสัญญาณรบกวนทั้ง 4 แบบ ที่ระดับ 25% 50% และ 100%

จากกราฟแท่ง แสดงเวลาที่ใช้ในขั้นตอนการทดสอบตัวแบบในข้อมูลคลื่นไฟฟ้าหัวใจที่มีสัญญาณรบกวนแบบต่าง ๆ ที่ระดับต่าง ๆ จะเห็นว่าโดยภาพรวม BOSSVS ใช้เวลาประมวลผลนานกว่า โดยเฉพาะเมื่อทดสอบกับข้อมูลคลื่นไฟฟ้าหัวใจที่มีสัญญาณรบกวนแบบ Powerline ที่ระดับ 100% ใช้เวลาสูงถึง 33 วินาที เนื่องจากสัญญาณรบกวนแบบ Powerline เกิดจากการรบกวนของสนามแม่เหล็กไฟฟ้า ซึ่งจำลองมาจากฟังก์ชันคลื่นไซน์ (Sine Wave) มีความถี่สูงประมาณ 50 เฮิรตซ์

4.3.12 การเปรียบเทียบค่าความถูกต้อง (Accuracy) คะแนน F1 (F1 Score) ค่าความแม่นยำ (Precision) และค่าความระลึก (Recall) ระหว่าง SAXVSM และ BOSSVS แบ่งตามสัญญาณรบกวนแต่ละแบบ ที่ระดับต่าง ๆ



The Accuracy Comparison

Noise Type	Level	Model: SAXVSM	Model: BOSSVS
None	0	0.99075	0.99075
baseline	25	0.98435	0.99075
	50	0.98791	0.99289
	100	0.99289	0.98009
composite	25	0.99004	0.99147
	50	0.99289	0.99004
	100	0.99218	0.98720
emg	25	0.99431	0.99004
	50	0.99147	0.99360
	100	0.98933	0.99218
powerline	25	0.98578	0.99147
	50	0.98933	0.98862
	100	0.97937	0.98933

The F1 Score Comparison

Noise Type	Level	Model: SAXVSM	Model: BOSSVS
None	0	0.98751	0.98749
baseline	25	0.97921	0.98758
	50	0.98389	0.99046
	100	0.99044	0.97297
composite	25	0.98654	0.98846
	50	0.99040	0.98654
	100	0.98949	0.98273
emg	25	0.99228	0.98654
	50	0.98846	0.99132
	100	0.98562	0.98941
powerline	25	0.98092	0.98846
	50	0.98570	0.98470
	100	0.97262	0.98554

The Precision Comparison

Noise Type	Level	Model: SAXVSM	Model: BOSSVS
None	0	0.98467	0.98654
baseline	25	0.96104	0.97917
	50	0.96828	0.98110
	100	0.98292	0.97485
composite	25	0.98464	0.98656
	50	0.98662	0.98464
	100	0.98106	0.97897
emg	25	0.99420	0.98464
	50	0.98656	0.99228
	100	0.98092	0.98846
powerline	25	0.97164	0.98656
	50	0.97547	0.97723
	100	0.95370	0.98649

The Recall Comparison

Noise Type	Level	Model: SAXVSM	Model: BOSSVS
None	0	0.99037	0.98844
baseline	25	0.99807	0.99615
	50	1.00000	1.00000
	100	0.99807	0.97110
composite	25	0.98844	0.99037
	50	0.99422	0.98844
	100	0.99807	0.98651
emg	25	0.99037	0.98844
	50	0.99037	0.99037
	100	0.99037	0.99037
powerline	25	0.99037	0.99037
	50	0.99615	0.99229
	100	0.99229	0.98459

ภาพที่ 16 Heat Map เปรียบเทียบค่าความถูกต้อง (Accuracy) (บนซ้าย) คะแนน F1 (F1 Score) (บนขวา) ค่าความแม่นยำ (Precision) (ล่างซ้าย) และค่าความระลึก (Recall) (ล่างขวา) ระหว่าง SAXVSM และ BOSSVS สำหรับข้อมูลคลื่นไฟฟ้าหัวใจที่มีสัญญาณรบกวนทั้ง 4 แบบ ที่ระดับ 25% 50% และ 100%

จาก Heat Map สามารถสรุปได้ว่า เมื่อไม่มีสัญญาณรบกวนทั้ง SAXVSM และ BOSSVS มีประสิทธิภาพดีใกล้เคียงกันอยู่ที่ประมาณ 98-99% และเมื่อมีสัญญาณรบกวนแบบต่าง ๆ ก็ยังคงมีประสิทธิภาพค่อนข้างคงที่

บทที่ 5

สรุปผล อภิปรายผลและข้อเสนอแนะ

5.1 สรุปผลการวิจัย

งานวิจัยนี้มีวัตถุประสงค์เพื่อเปรียบเทียบประสิทธิภาพการจำแนกประเภทข้อมูลคลื่นไฟฟ้าหัวใจเต้นผิดจังหวะที่มีสัญญาณรบกวนแบบ EMG Powerline Baseline และ Composite ที่ระดับ 25% 50% และ 100% ด้วยวิธี SAXVSM และ BOSSVS โดยจะเปรียบเทียบในเรื่องของเวลาและประสิทธิภาพของการจำแนกประเภทข้อมูล ซึ่งสามารถสรุปผลการวิจัยได้ดังนี้

สำหรับข้อมูลคลื่นไฟฟ้าหัวใจที่ไม่มีสัญญาณรบกวน ทั้งสองตัวแบบให้ค่าความถูกต้อง (Accuracy) อยู่ที่ประมาณ 99% แต่เมื่อเพิ่มสัญญาณรบกวนเข้าไปทำให้ค่าความถูกต้อง (Accuracy) ของทั้งสองตัวแบบลดลงเพียงเล็กน้อยแต่ยังคงทำงานได้ดีใกล้เคียงกัน มีค่าความถูกต้อง (Accuracy) ประมาณ 97-99%

สำหรับข้อมูลคลื่นไฟฟ้าหัวใจที่ไม่มีสัญญาณรบกวน ทั้งสองตัวแบบให้คะแนน F1 (F1 Score) อยู่ที่ประมาณ 99% แต่เมื่อเพิ่มสัญญาณรบกวนเข้าไปทำให้คะแนน F1 (F1 Score) ทั้งสองตัวแบบลดลงเพียงเล็กน้อยแต่ยังคงทำงานได้ดีใกล้เคียงกัน มีคะแนน F1 (F1 Score) ประมาณ 97-99%

สำหรับข้อมูลคลื่นไฟฟ้าหัวใจที่ไม่มีสัญญาณรบกวน ทั้งสองตัวแบบให้ค่าความแม่นยำ (Precision) อยู่ที่ประมาณ 98% แต่เมื่อเพิ่มสัญญาณรบกวนเข้าไปทำให้ค่าความแม่นยำ (Precision) ทั้งสองตัวแบบลดลงเพียงเล็กน้อยแต่ยังคงทำงานได้ดีใกล้เคียงกัน มีค่าความแม่นยำ (Precision) ประมาณ 95-99%

สำหรับข้อมูลคลื่นไฟฟ้าหัวใจที่ไม่มีสัญญาณรบกวน ทั้งสองตัวแบบให้ค่าความระลึก (Recall) อยู่ที่ประมาณ 99% แต่เมื่อเพิ่มสัญญาณรบกวนเข้าไปทำให้ค่าความระลึก (Recall) ทั้งสองตัวแบบลดลงเพียงเล็กน้อยแต่ยังคงทำงานได้ดีใกล้เคียงกัน มีค่าความระลึก (Recall) ประมาณ 97-100%

สำหรับเวลาในการประมวลผลของ SAXVSM และ BOSSVS ทั้งในส่วนขั้นตอนการสอนตัวแบบด้วยการใช้ 10-Fold Cross Validation เพื่อเลือกพารามิเตอร์ และขั้นตอนการทดสอบตัวแบบ เพื่อเปรียบเทียบประสิทธิภาพ เนื่องจาก BOSSVS มีกระบวนการที่ซับซ้อนมากกว่า SAXVSM จึงใช้เวลาในการประมวลผลนานกว่า

สำหรับผลการเปรียบเทียบค่าความถูกต้อง (Accuracy) คะแนน F1 (F1 Score) ค่าความแม่นยำ (Precision) และค่าความระลึก (Recall) ระหว่างตัวแบบ SAXVSM และ BOSSVS สำหรับข้อมูลคลื่นไฟฟ้าหัวใจเต้นผิดจังหวะที่มีสัญญาณรบกวนแบบ EMG Powerline Baseline และ Composite ที่ระดับ 25% 50% และ 100% สามารถแสดงได้ดังตารางต่อไปนี้

ตารางที่ 27 ผลการเปรียบเทียบค่าความถูกต้อง (Accuracy) ในข้อมูลทั้ง 13 ชุด

No.	Data	Winning model of Accuracy comparison	
		SAXVSM	BOSSVS
1.	ECG	tie	tie
2.	data_emg_noise25	1	0
3.	data_emg_noise50	0	1
4.	data_emg_noise100	0	1
5.	data_power_noise25	0	1
6.	data_power_noise50	1	0
7.	data_power_noise100	0	1
8.	data_base_noise25	0	1
9.	data_base_noise50	0	1
10.	data_base_noise100	1	0
11.	data_comb_noise25	0	1
12.	data_comb_noise50	1	0
13.	data_comb_noise100	1	0
Win		5	7

โดยที่ 1 หมายถึง ตัวแบบนั้นให้ผลความถูกต้อง (Accuracy) สูงกว่า (Win)

0 หมายถึง ตัวแบบนั้นให้ผลความถูกต้อง (Accuracy) ต่ำกว่า (Loss)

Tie หมายถึง ทั้ง 2 ตัวแบบให้ผลความถูกต้อง (Accuracy) เท่ากัน

เมื่อเปรียบเทียบค่าความถูกต้อง (Accuracy) สำหรับข้อมูลคลื่นไฟฟ้าหัวใจทั้งหมด 13 ชุด พบว่า โดยส่วนใหญ่ตัวแบบ BOSSVS มีค่าความถูกต้อง (Accuracy) มากกว่า

ตารางที่ 28 ผลการเปรียบเทียบคะแนน F1 (F1 Score) ในข้อมูลทั้ง 13 ชุด

No.	Data	Winning model of F1 Score comparison	
		SAXVSM	BOSSVS
1.	ECG	1	0
2.	data_emg_noise25	1	0
3.	data_emg_noise50	0	1
4.	data_emg_noise100	0	1
5.	data_power_noise25	0	1

6.	data_power_noise50	1	0
7.	data_power_noise100	0	1
8.	data_base_noise25	0	1
9.	data_base_noise50	0	1
10.	data_base_noise100	1	0
11.	data_comb_noise25	0	1
12.	data_comb_noise50	1	0
13.	data_comb_noise100	1	0
Win		6	7

โดยที่ 1 หมายถึง ตัวแบบนั้นให้คะแนน F1 (F1 Score) สูงกว่า (Win)

0 หมายถึง ตัวแบบนั้นให้คะแนน F1 (F1 Score) ต่ำกว่า (Loss)

Tie หมายถึง ทั้ง 2 ตัวแบบให้คะแนน F1 (F1 Score) เท่ากัน

เมื่อเปรียบเทียบคะแนน F1 (F1 Score) สำหรับข้อมูลคลื่นไฟฟ้าหัวใจทั้งหมด 13 ชุด พบว่าโดยส่วนใหญ่ตัวแบบ BOSSVS มีคะแนน F1 (F1 Score) มากกว่า

ตารางที่ 29 ผลการเปรียบเทียบค่าความแม่นยำ (Precision) ในข้อมูลทั้ง 13 ชุด

No.	Data	Winning model of Precision comparison	
		SAXVSM	BOSSVS
1.	ECG	0	1
2.	data_emg_noise25	1	0
3.	data_emg_noise50	0	1
4.	data_emg_noise100	0	1
5.	data_power_noise25	0	1
6.	data_power_noise50	0	1
7.	data_power_noise100	0	1
8.	data_base_noise25	0	1
9.	data_base_noise50	0	1
10.	data_base_noise100	1	0
11.	data_comb_noise25	0	1
12.	data_comb_noise50	1	0
13.	data_comb_noise100	1	0
Win		4	9

โดยที่ 1 หมายถึง ตัวแบบนั้นให้ค่าความแม่นยำ (Precision) สูงกว่า (Win)

0 หมายถึง ตัวแบบนั้นให้ค่าความแม่นยำ (Precision) ต่ำกว่า (Loss)

Tie หมายถึง ทั้ง 2 ตัวแบบให้ค่าความแม่นยำ (Precision) เท่ากัน

เมื่อเปรียบเทียบค่าความแม่นยำ (Precision) สำหรับข้อมูลคลื่นไฟฟ้าหัวใจทั้งหมด 13 ชุด พบว่าตัวแบบ BOSSVS มีค่าความแม่นยำ (Precision) มากกว่าเป็นส่วนใหญ่

ตารางที่ 30 ผลการเปรียบเทียบค่าความระลึก (Recall) ในข้อมูลทั้ง 13 ชุด

No.	Data	Winning model of Recall comparison	
		SAXVSM	BOSSVS
1.	ECG	1	0
2.	data_emg_noise25	1	0
3.	data_emg_noise50	tie	tie
4.	data_emg_noise100	tie	tie
5.	data_power_noise25	tie	tie
6.	data_power_noise50	1	0
7.	data_power_noise100	1	0
8.	data_base_noise25	1	0
9.	data_base_noise50	tie	tie
10.	data_base_noise100	1	0
11.	data_comb_noise25	0	1
12.	data_comb_noise50	1	0
13.	data_comb_noise100	1	0
Win		8	1

โดยที่ 1 หมายถึง ตัวแบบนั้นให้ค่าความระลึก (Recall) สูงกว่า (Win)

0 หมายถึง ตัวแบบนั้นให้ค่าความระลึก (Recall) ต่ำกว่า (Loss)

Tie หมายถึง ทั้ง 2 ตัวแบบให้ค่าความระลึก (Recall) เท่ากัน

เมื่อเปรียบเทียบค่าความระลึก (Recall) สำหรับข้อมูลคลื่นไฟฟ้าหัวใจทั้งหมด 13 ชุด พบว่าโดยส่วนใหญ่ตัวแบบ SAXVSM มีค่าความระลึก (Recall) มากกว่า

ตารางที่ 31 ผลการเปรียบเทียบเวลาในการประมวลผล ในข้อมูลทั้ง 13 ชุด

No.	Data	Winning model of Running Time comparison	
		SAXVSM	BOSSVS
1.	ECG	1	0
2.	data_emg_noise25	1	0
3.	data_emg_noise50	0	1
4.	data_emg_noise100	0	1
5.	data_power_noise25	1	0
6.	data_power_noise50	1	0
7.	data_power_noise100	1	0
8.	data_base_noise25	0	1
9.	data_base_noise50	1	0
10.	data_base_noise100	1	0
11.	data_comb_noise25	1	0
12.	data_comb_noise50	1	0
13.	data_comb_noise100	1	0
Win		10	3

โดยที่ 1 หมายถึง ตัวแบบนั้นใช้เวลาประมวลผลน้อยกว่า (Win)

0 หมายถึง ตัวแบบนั้นใช้เวลาประมวลผลมากกว่า (Loss)

เมื่อเปรียบเทียบเวลาที่ใช้ในการประมวลผลสำหรับข้อมูลคลื่นไฟฟ้าหัวใจทั้งหมด 13 ชุด พบว่า โดยส่วนใหญ่ตัวแบบ SAXVSM ใช้เวลาในประมวลผลน้อยกว่า

5.2 อภิปรายผลการวิจัย

การเปรียบเทียบประสิทธิภาพของการจำแนกประเภทข้อมูลอนุกรมเวลาระหว่าง SAXVSM และ BOSSVS โดยใช้ข้อมูลคลื่นไฟฟ้าฟัวใจเป็นกรณีศึกษา และทำการเพิ่มสัญญาณรบกวนแบบ EMG Powerline และ Baseline ที่ระดับ 25% 50% และ 100% สามารถสรุปได้ว่า โดยภาพรวมทั้ง 2 ตัวแบบนี้มีประสิทธิภาพดีใกล้เคียงกัน ทั้งในแง่ของความถูกต้อง คะแนน F1 ค่าความแม่นยำ และค่าความระลึก โดยเฉลี่ยอยู่ที่ 98-99% สำหรับข้อมูลทั้ง 13 ชุด ทั้งข้อมูลคลื่นไฟฟ้าหัวใจที่ยังไม่ได้มีการเติมสัญญาณรบกวนหรือมีการเพิ่มสัญญาณรบกวนแบบต่าง ๆ อย่างไรก็ดี เมื่อเปรียบเทียบเวลาที่ใช้ในการประมวลผล เนื่องจาก SAXVSM มีกระบวนการที่ซับซ้อนน้อยกว่าจะใช้เวลาน้อยกว่า BOSSVS ทั้งในขั้นตอนการสอนตัวแบบและการทดสอบตัวแบบ

งานวิจัยของ Senin & Malinchik (2013) ได้เปรียบเทียบผลกระทบต่อประสิทธิภาพของตัวแบบเมื่อเพิ่มสัญญาณรบกวนแบบแบบเก๊าซเซี่ยล (Gaussian noise) โดยใช้ข้อมูลอนุกรมเวลา Cylinder-Bell-Funnel (CBF) และวัดอัตราความคลาดเคลื่อนระหว่างตัวแบบจำแนกประเภทต่าง ๆ พบว่าประสิทธิภาพการจำแนกของตัวแบบ SAX ค่อนข้างคงที่ ไม่ค่อยมีผลกระทบต่อสัญญาณรบกวนในระดับต่ำ ๆ เมื่อเปรียบเทียบกับ 1-NN Euclidean เช่นเดียวกับงานวิจัยของ Schäfer (2014, 2015) ที่ได้ใช้ข้อมูลนี้และเปรียบเทียบกับ 1NN-Dynamic Time Wraping (DTW) และ 1-NN Euclidean ได้ผลดีเช่นเดียวกัน สำหรับประสิทธิภาพของตัวแบบ SAXVSM และ BOSSVS ที่ใช้ทดสอบกับข้อมูล ECG5000 ในงานวิจัยนี้ก็เป็นไปตามงานวิจัยที่ผ่านมา

สำหรับตัวแบบ SAXVSM เมื่อพิจารณาถึงค่าความถูกต้อง (Accuracy) คะแนน F1 (F1 Score) และค่าความแม่นยำ (Precision) ที่ทดสอบกับข้อมูลที่มีสัญญาณรบกวนแบบ Powerline ที่ระดับ 100% ทำให้ตัววัดประสิทธิภาพทั้ง 3 ตัวข้างต้นน้อยที่สุด ซึ่งสัญญาณรบกวนประเภทนี้เป็นคลื่นความถี่สูง เกิดจากการรบกวนของสนามแม่เหล็กไฟฟ้า ถูกจำลองมาจากฟังก์ชันคลื่นไซน์ (Sine Wave) ยิ่งที่ระดับสัญญาณรบกวนที่สูงขึ้น อาจมีส่วนทำให้ระยะขจัดที่แกว่งจากแนวเส้นฐาน (Amplitude) ผิดเพี้ยนไปจากรูปแบบของคลื่นไฟฟ้าหัวใจปกติ ทำให้ตัวแบบทำนายผิดพลาดได้ อย่างไรก็ดี เมื่อพิจารณาในส่วนของเวลาในการประมวลผลทั้งขั้นตอนการสอนและการทดสอบตัวแบบ SAXVSM รวดเร็วกว่า BOSSVS อย่างเห็นได้ชัด

สำหรับตัวแบบ BOSSVS เมื่อพิจารณาถึงค่าความถูกต้อง (Accuracy) คะแนน F1 (F1 Score) ค่าความแม่นยำ (Precision) และค่าความระลึก (Recall) ที่ทดสอบกับข้อมูลที่มีสัญญาณรบกวนแบบ Baseline ที่ระดับ 100% ทำให้ตัววัดประสิทธิภาพทั้ง 4 ตัวข้างต้นน้อยที่สุด ซึ่งสัญญาณรบกวนประเภทนี้เกิดขึ้นได้จากสาเหตุทางกายภาพ เช่น การเคลื่อนไหวของร่างกาย ความชื้นหรือเหงื่อที่ผิวหนัง การเคลื่อนที่ของอิเล็กโทรดหรือขั้วไฟฟ้าที่ติดบนผิวหนังและเชื่อมเข้ากับเครื่องบันทึกคลื่นไฟฟ้าหัวใจ ถึงแม้จะมีความถี่ต่ำแต่มักพบสัญญาณรบกวนประเภทนี้ได้ง่าย สัญญาณรบกวน

ประเภทนี้ถูกจำลองมาจากฟังก์ชันคลื่นไซน์ (Sine Wave) ยิ่งที่ระดับสัญญาณรบกวนที่สูงขึ้น อาจจะมีส่วนทำให้ระยะขจัดที่แกว่งจากแนวเส้นฐาน (Amplitude) ผิดเพี้ยนไปจากรูปแบบของคลื่นไฟฟ้าหัวใจปกติ ผู้วิจัยสันนิษฐานว่าในขณะที่อาสาสมัครเข้ารับการตรวจคลื่นไฟฟ้าหัวใจ แค่มีการขยับตัวเพียงเล็กน้อยก็เกิดสัญญาณรบกวนประเภทนี้เข้ามาสอดแทรกได้ ทำให้ลักษณะของคลื่นผิดเพี้ยนไปในบางช่วง ทำให้ทั้งตัววัดประสิทธิภาพทั้ง 4 ค่าให้ผลออกมาน้อยที่สุดเมื่อเทียบกับสัญญาณรบกวนแบบอื่น เมื่อพิจารณาค่าความระลึกเมื่อทดสอบกับสัญญาณรบกวนแบบ Baseline ที่ระดับ 50% ตัวแบบไม่มีการทำนายผิดพลาด แต่เมื่อระดับ 100% ค่าความระลึก (Recall) ลดลงน้อยที่สุด เมื่อเทียบกับสัญญาณรบกวนแบบอื่น ๆ จึงเป็นข้อควรระวังหากสามารถตรวจจับสัญญาณรบกวนประเภทนี้ได้ก็ควรใช้เทคนิคการกรองสัญญาณรบกวนประเภทนี้ออก เนื่องจากตัวแบบอาจจะยังทำงานได้ไม่ดีเท่าที่ควร

สำหรับเวลาที่ใช้ในการประมวลผลสำหรับตัวแบบ SAXVSM และ BOSSVS ในขั้นตอนการเลือกพารามิเตอร์โดยใช้ 10-Fold Cross Validation สำหรับข้อมูลในแต่ละชุด ยิ่งมีการเพิ่มสัญญาณรบกวนในระดับที่สูงขึ้นทั้ง SAXVSM และ BOSSVS จะใช้เวลาในการสอนตัวแบบน้อยลง เนื่องจากในขั้นตอนการสร้างเมทริกซ์น้ำหนักคำด้วยการหาความถี่ของคำและการผกผันความถี่ในเอกสาร (TF-IDF) ยิ่งข้อมูลที่มีระดับสัญญาณรบกวนสูงขึ้น ทำให้ตัวแบบเจอรูปแบบของสัญญาณรบกวนมาบดบังรูปแบบที่แท้จริงของคลื่นไฟฟ้าหัวใจ ทำให้ได้จำนวนรูปแบบตัวอักษรระหว่างประเภทที่ปกติและประเภทผิดปกติซ้ำกันมากขึ้น และความหลากหลายของรูปแบบตัวอักษรน้อยลง จึงทำให้เมทริกซ์มีขนาดเล็กลง เมื่อไปคำนวณค่าความเหมือนโคไซน์ (Cosine Similarity) จึงทำได้รวดเร็วขึ้น

โดยสรุปจากการวิจัยนี้ ทั้งสองตัวแบบนี้เป็นตัวแบบการจำแนกประเภทข้อมูลอนุกรมเวลาที่ใช้เทคนิคการลดมิติของอนุกรมเวลาด้วยการแบ่งอนุกรมเวลาเป็นอนุกรมเวลาย่อย ๆ และพยายามแปลงให้เป็นลำดับของตัวอักษรที่สามารถอธิบายได้ด้วย Piecewise Aggregate Approximation (PAA) สำหรับ SAX และ Discrete Fourier Transform (DFT) สำหรับ BOSS จะได้เมทริกซ์ความถี่ของคำ (Term Frequency Matrix) จากนั้นคำนวณน้ำหนักของคำที่อยู่ในประเภท (Class) ต่าง ๆ ด้วยการหาความถี่ของคำและการผกผันความถี่ในเอกสาร (Term Frequency-Inverse Document Frequency: TF-IDF) เนื่องจากการคำนวณเพียงความถี่อาจจะไม่เพียงพอต่อการระบุความสำคัญของรูปแบบตัวอักษรนั้น ๆ จึงต้องพิจารณาความถี่ของรูปแบบของอักษรนั้นแต่จะถูกลดทอนลงด้วยสัดส่วนของรูปแบบตัวอักษรนั้นที่อยู่ในประเภท (Class) อื่น ๆ ด้วย และหาค่าความเหมือนโคไซน์ (Cosine Similarity) เพื่อใช้ในการจำแนกประเภทข้อมูลต่อไป ผลการวิจัยพบว่าทั้ง 2 ตัวแบบเหมาะสำหรับการจำแนกประเภทข้อมูลอนุกรมเวลา และมีประสิทธิภาพดีใกล้เคียงกัน แต่ SAXVSM ใช้เวลาการประมวลผลน้อยกว่า

5.3 ข้อเสนอแนะ

งานวิจัยนี้ใช้ข้อมูลคลื่นไฟฟ้าหัวใจจากฐานข้อมูล Physionet เพื่อเปรียบเทียบประสิทธิภาพของตัวแบบจำแนกประเภทข้อมูลอนุกรมเวลา ซึ่งให้ผลการเปรียบเทียบประสิทธิภาพของทั้ง 2 วิธียังไม่ชัดเจนเท่าที่ควร อาจเพราะปริมาณข้อมูลที่นำมาศึกษาน้อยเกินไป ในการวิจัยครั้งต่อไปควรเพิ่มปริมาณข้อมูลที่ใช้ศึกษาให้มากขึ้น รวมทั้งศึกษาข้อมูลด้านอื่น ๆ เพิ่มเติมด้วย อีกทั้งงานวิจัยนี้ทำการเปรียบเทียบประสิทธิภาพของตัวแบบจำแนกประเภทข้อมูลอนุกรมเวลาด้วยวิธีการ 2 วิธี ได้แก่ SAXVSM และ BOSSVS เท่านั้น ในการวิจัยครั้งต่อไปอาจจะเพิ่มการเปรียบเทียบการจำแนกประเภทข้อมูลอนุกรมเวลาด้วยเทคนิคอื่น ๆ เพิ่มเติม

บรรณานุกรม

av Fredrik Edin. (2020). *Time series classification – an overview*. https://developersbay.se/time-series-classification-an-overview/

Biosense Webster. (2020). *“หัวใจเต้นผิดจังหวะ” ภัยเงียบต้องรู้ คาดปี 93 มีผู้ป่วย 72 ล้านคนทั่วภูมิภาค*. https://www.bangkokbiznews.com/news/detail/852765

Chang, K. M. (2010). Arrhythmia ECG noise reduction by ensemble empirical mode decomposition. *Sensors (Basel)*, *10*(6), 6063-6080. https://doi.org/10.3390/s100606063

Gatchalee, P. (2019). *Confusion Matrix เครื่องมือสำคัญในการประเมินผลลัพธ์ของการทำนายใน Machine learning*. https://bit.ly/3pm2Lxq

Kaya, Y., & Pehlivan, H. (2015). Classification of premature ventricular contraction in ECG. *Int J Adv Comput Sci Appl*, *6*(7), 34-40. https://doi.org/10.14569/IJACSA.2015.060706

Lin, J., Keogh, E., Wei, L., & Lonardi, S. (2007). Experiencing SAX: a novel symbolic representation of time series. *Data Mining and Knowledge Discovery*, *15*(2), 107-144. https://doi.org/10.1007/s10618-007-0064-z

Medthai. (2018). *การตรวจคลื่นไฟฟ้าหัวใจ (Electrocardiogram : ECG หรือ EKG)*. https://bit.ly/3Ecfk4u

MIT Laboratory for Computational Physiology. (1999). *PhysioNet*. https://physionet.org/

Pathangay, V., & Rath, S. P. (2014). Arrhythmia detection in single-lead ECG by combining beat and rhythm-level information. *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 3236-3239. https://doi.org/ 10.1109/EMBC.2014.6944312

Raghavan, V. V., & Wong, S. M. (1986). A critical analysis of vector space model for information retrieval. *Journal of the American Society for information Science*, *37*(5), 279-287. https://doi.org/10.1002/(SICI)1097-4571(198609)37:5<279::AID-ASI1>3.0.CO;2-Q

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information processing & management*, *24*(5), 513-523.

https://doi.org/10.1016/0306-4573(88)90021-0

Schäfer, P. (2014). The BOSS is concerned with time series classification in the presence of noise. *Data Mining and Knowledge Discovery*, *29*(6), 1505-1530. https://doi.org/10.1007/s10618-014-0377-7

Schäfer, P. (2015a). *Bag-Of-SFA-Symbols in Vector Space (BOSS VS)*.

Schäfer, P. (2015b). Scalable time series similarity search for data analytics. *Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät*. https://doi.org/10.18452/17338

Souspace. (2020). *รู้จัก Discrete Fourier Transform และ Fast Fourier Transform*. https://www.youtube.com/watch?v=a03-5mr5yn4

Srikong, N. (2020). *Cosine similarity*. https://bit.ly/3od2xJC

Thailand Online Hospital. (2016). *การตรวจคลื่นไฟฟ้าหัวใจ (Electrocardiography)*. https://bit.ly/3EiRCUa

ขนิษฐา ดีสุบิน. (2560). การพัฒนาตัวแบบการพยากรณ์ความถนัดทางการเรียนตามทฤษฎี 4 MAT โดยการวิเคราะห์ด้วยวิธีต้นไม้. *วารสารนวัตกรรมการเรียนรู้*, 3(1), 15. https://doi.org/10.14416/j.fte.2016.6.0015

ธรรมศักดิ์ เธียรนิเวศน์. (2548). *การลดขนาดข้อมูลด้วยน้ำหนักความหนาแน่นเพื่อการจัดกลุ่มข้อมูลขนาดใหญ่* https://bit.ly/31qh2k7

ปริวัตร เพ็งแก้ว. (2563). *จบปัญหาหัวใจเต้นผิดจังหวะ ด้วยการสวนและจี้ด้วยคลื่นวิทยุไฟฟ้าหัวใจ*. https://www.naewna.com/lady/510649

ภูมิฐาน รังคกูลนุวัฒน์. (2562). *การวิเคราะห์อนุกรมเวลาสำหรับเศรษฐศาสตร์และธุรกิจ*. https://economics.utcc.ac.th/wp-content/uploads/Time-Series-for-Econ-and-Bus_Poomthan.pdf

ภาคผนวก

การวิจัยนี้ใช้โปรแกรม Python3.6.13 ในการจำลองและการวิเคราะห์ข้อมูล

```
# Import Library
import numpy as np
import pandas as pd
from scipy.io import arff
import plotly.express as px
import matplotlib.pyplot as plt
import itertools
import random
from random import sample
from collections import Counter
import seaborn as sns
import os
from pyts.classification import SAXVSM
from pyts.classification import BOSSVS
from pyts.transformation import BOSS
from sklearn.model_selection import train_test_split, GridSearchCV, cross_validate,
cross_val_score, KFold, RandomizedSearchCV
```



```
# Import File (.arff)
data = arff.loadarff(' ## File path ## ')
df = pd.DataFrame(data[0])
df.loc[:,['target']] = df.loc[:,['target']].astype(int)
df['target'] = df['target'] - 1
df = df.sample(frac = 1, random_state = 1000)
data_test= arff.loadarff(' ## File path ## ')
df2 = pd.DataFrame(data_test[0])
df2.loc[:,['target']] = df2.loc[:,['target']].astype(int)
df2['target'] = df2['target'] -1
```

```
df2 = df2.sample(frac = 1, random_state = 1000)
data = pd.concat([df,df2])
data['target'].value_counts()
data_0_1 = data [ (data ['target'] == 0 )  | (data ['target'] == 1 ) ]

## Define Function: "Kfold_model_score" to return Mean Accuracy in K-Fold Cross Validation for SAXVSM or BOSSVS
# Input: data, model_str, window_size(w), word_size(p), alphabet_size(a)
# Output: Mean Accuracy
def Kfold_model_score ( data, model_str, w, p ,a ) :
    X = data.iloc[:,:-1].values
    y = data.iloc[:,-1].values
    X_train, X_test, y_train, y_test = train_test_split( X, y, test_size = 0.3 , random_state = 1000)
    kf = KFold (n_splits = 10 , random_state= 1000, shuffle = True)
    score = []
    errl = []
    list_pattern = []
    tfidf_dict_list = []
    model_str = model_str.lower()
    if model_str == 'sax' or model_str == 'saxvsm' :
        model =  SAXVSM ( window_size = w, word_size = p,  n_bins = a, strategy='normal')
    elif model_str == 'boss' or model_str == 'bossvs' :
        model =  BOSSVS ( window_size = w, word_size = p,  n_bins = a, strategy='quantile')
    try :
        for train_index, test_index in (kf.split(X_train)):
            model.fit ( X_train[train_index] , y_train[train_index] )
            tfidf = model.tfidf_
            keys = model.vocabulary_.values()
```

```
            tfidf_dict_list = [{key : val for key,  val in zip(keys, idx)} for idx in tfidf]
            list_pattern.append(tfidf_dict_list)
            X_new = model.decision_function(X_train[test_index])
            y_pred = model.predict(X_train[test_index])
            err = np.size(np.where( (y_train[test_index] != y_pred) ) ) / np.size(
y_train[test_index]) # Equivalent to model.score()
            errl.append(err)
            model_score = model.score (X_train[test_index] , y_train[test_index] )
            score.append(model_score)
        mean_model_score = np.mean(score)
        sd_model = np.std(score)
    except UnboundLocalError :
        mean_model_score = 'Incorrect Model'
    return mean_model_score, sd_model
# Call Function: "Kfold_model_score"
# SAX: Kfold_model_score( data = data_0_1 , model_str = 'SAX', w = 30, p =10, a =4
)
# BOSS: Kfold_model_score( data = data_0_1 , model_str = 'BOSS', w = 30, p =10, a
=4 )
```



```
## Define Function: "Kfold_tfidf" to return TF-IDF Matrix in K-Fold Cross Validation for
SAXVSM or BOSSVS
# Input: data, model_str, window_size(w), word_size(p), alphabet_size(a)
# Output: TF-IDF 0f Normal_class(0) and TF-IDF 0f Abnormal_class(1)
def Kfold_tfidf ( data, model_str, w, p, a ) :
    list_vs = []
    X = data.iloc[:,:-1].values
    y = data.iloc[:,-1].values
    X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.3 ,
random_state= 1000)
    kf = KFold (n_splits = 10 , random_state = 1000, shuffle = True)
```

```
    model_str = model_str.lower()
    if model_str == 'sax' or model_str == 'saxvsm' :
        model =  SAXVSM ( window_size = w, word_size = p,  n_bins = a,
strategy='normal')
    elif model_str == 'boss' or model_str == 'bossvs' :
        model =  BOSSVS ( window_size = w, word_size = p,  n_bins = a,
strategy='quantile')
    try :
        for train_index, test_index in (kf.split(X_train)):
            model =  BOSSVS ( window_size = w , word_size = p,  n_bins = a ,
strategy='quantile')
            model. fit ( X_train[train_index] , y_train[train_index] )
            tfidf = model.tfidf_
            keys = model.vocabulary_.values()
            temp = [{key : val for key,  val in zip(keys, idx)} for idx in tfidf]
            X_new = model.decision_function(X_train[test_index])
            cosine_sim = X_new
            y_pred = model.predict(X_train[test_index])
            print(f'train_shape: {train_index.shape}, validate_shape: {test_index.shape}')
            print(f'cosine_similarity : {X_new}')
            print(f' Shape of Cosine_Similarity: {X_new.shape}')
            print(f' predicted : {(y_pred)}')
            print(f' y_true    : {y_train[test_index]} ')
            model_score = model . score ( X_train[test_index] , y_train[test_index] )
            print(f' {model_str}_score : {model_score} ' )
            acc = np.size(np.where( (y_train[test_index] == y_pred) ) ) / np.size(
y_train[test_index])  # Equivalent to model.score()
            print(f' accuracy : {acc}')
            list_vs.append(temp)
        model_df = pd.DataFrame(list_vs)
        normal_tfidf = model_df[0].apply(pd.Series)
```

```
        normal_tfidf = normal_tfidf.append(normal_tfidf.mean(axis=0) , ignore_index=
True)
        normal_tfidf = normal_tfidf.sort_values(normal_tfidf.last_valid_index(), axis=1,
ascending= False  )
        normal_tfidf = normal_tfidf.reset_index()
        normal_tfidf = normal_tfidf.rename(columns = {'index' : 'Fold'})
        normal_tfidf['Fold'] = normal_tfidf['Fold'].replace(10 , 'Mean of TF-IDF in 10-
Fold')
        abnormal_tfidf = model_df[1].apply(pd.Series)
        abnormal_tfidf = abnormal_tfidf.append(abnormal_tfidf.mean(axis=0) ,
ignore_index= True)
        abnormal_tfidf = abnormal_tfidf.sort_values(abnormal_tfidf.last_valid_index(),
axis=1, ascending= False  )
        abnormal_tfidf = abnormal_tfidf.reset_index()
        abnormal_tfidf = abnormal_tfidf.rename(columns = {'index' : 'Fold'})
        abnormal_tfidf['Fold'] = abnormal_tfidf['Fold'].replace(10 , 'Mean of TF-IDF in 10-
Fold')
    except UnboundLocalError :
        normal_tfidf = 'Incorrect Model'
        abnormal_tfidf = 'Incorrect Model'
    return normal_tfidf , abnormal_tfidf , cosine_sim
# Call Function: "Kfold_tfidf"
# SAX: Kfold_tfidf( data = data_0_1 , model_str = 'SAX', w = 30, p =10, a = 4 )
# BOSS: Kfold_tfidf( data = data_0_1 , model_str = 'BOSS', w = 30, p =10, a = 4 )

## Define Function: "allparam_allscore_model" to return All parameters and their
accuracy with randomly n_sample
# Input: data , model_str,  lower_w, upper_w , lower_p, upper_p,  lower_a, upper_a ,
n_sample
# Output: random_param_score is the parameters
```

```
def allparam_allscore_model ( data , model_str,  lower_w, upper_w , lower_p,
upper_p,  lower_a, upper_a , n_sample ) :
    random.seed(3061996)
    param = [ np.arange( lower_w, upper_w).tolist() , np.arange(lower_p,
upper_p).tolist(),  np.arange(lower_a, upper_a).tolist() ]
    comb_param = list(itertools.product(*param))
    param_list = []
    random_param_score = []
    for i in comb_param :
        if (i[0] > i[1]) & (i[1] >= i[2]) :
            param_list.append(i)
    sample_comb_param = sample(param_list, n_sample)
    model_str = model_str.lower()
    #print(model_str)
    if model_str == 'sax' or model_str == 'saxvsm' or model_str == 'boss' or model_str
== 'bossvs' :
        for index, param_idx in enumerate(sample_comb_param) :
            dict_param = {}
            model_score , sd_model = Kfold_model_score (data, model_str,
param_idx[0], param_idx[1], param_idx[2])
            print(f' round {index} : w : {param_idx[0]} | p : {param_idx[1]} |  a :
{param_idx[2]}   -------->   {model_score} , {sd_model} ')
            dict_param['window'] = param_idx[0]
            dict_param['word'] = param_idx[1]
            dict_param['alphabet'] = param_idx[2]
            dict_param['accuracy'] = model_score
            dict_param['SD'] = sd_model
            random_param_score.append(dict_param)
    else :
        random_param_score = 'Incorrect Model'
    return random_param_score
```

```
# Call Function: "allparam_allscore_model"
# SAX: Kfold_tfidf( data = data_0_1 , lower_w = 10 , upper_w = 50 , lower_p = 3 ,
upper_p = 20 ,  lower_a = 2, upper_a = 10 , n_sample = 30)
# BOSS: Kfold_tfidf( data = data_0_1 , lower_w = 10 , upper_w = 50 , lower_p = 3 ,
upper_p = 20 ,  lower_a = 2, upper_a = 10 , n_sample = 30)

## Define Function: "data_emg_noise" to return ECG + EMG Noises
# Input: percentage of noises
# Output: ECG + EMG Noises (DataFrame)
def data_emg_noise (percent) :
    mu , sigma =  0, 1
    np.random.seed(3061996)
    random_nums = np.random.normal(mu , sigma  , 140)
    Vpp = 5
    reduced_ratio = 1/8
    noise = random_nums*(Vpp)*(reduced_ratio)*(percent/100)
    data_emg = data_0_1.iloc[:,:-1] + noise
    label = data_0_1.iloc[:,-1]
    data_emg = pd.concat( [data_emg, label ] ,axis =1 )
    return data_emg
# Call Function: "data_emg_noise"
# data_emg_noise (percent = 25)

## Define Function: "data_powerlines_noise" to return ECG + Powerlines Noises
# Input: percentage of noises
# Output: ECG + Powerlines Noises (DataFrame)
def data_powerlines_noise (percent) :
    np.random.seed(3061996)
    x1 = np.random.uniform(-5, 5)
    x2 = np.random.uniform(-5, 5)
    random_num = np.linspace(min(x1,x2), max(x1, x2), 140)
```

```
    reduced_ratio = 1/4
    Vpp = 5
    freq = 50
    power_noise = (np.sin(2*np.pi* random_num *
freq)*(Vpp*(percent/100)*(reduced_ratio)))
    data_power = data_0_1.iloc[:,:-1] + power_noise
    label = data_0_1.iloc[:,-1]
    data_power = pd.concat( [data_power, label ] ,axis =1 )
    return data_power
# Call Function: "data_powerlines_noise"
# data_powerlines_noise (percent = 25)

## Define Function: "data_baselines_noise" to return ECG + Baselines Noises
# Input: percentage of noises
# Output: ECG + Baselines Noises (DataFrame)
def data_baselines_noise (percent) :
    np.random.seed(3061996)
    x1 = np.random.uniform(0, 10)
    x2 = np.random.uniform(0, 10)
    random_num = np.linspace(min(x1,x2), max(x1, x2), 140)
    Vpp = 5
    freq = 0.333
    baseline_noise = (np.sin(2*np.pi* random_num * freq)*(Vpp*(percent/100)))
    data_base = data_0_1.iloc[:,:-1] + baseline_noise
    label = data_0_1.iloc[:,-1]
    data_base = pd.concat( [data_base, label ] ,axis =1 )
    return data_base
# Call Function: "data_baselines_noise"
# data_baselines_noise (percent = 25)

## Define Function: "data_composite_noise" to return ECG + Composite Noises
```

```
# Input: percentage of noises
# Output: ECG + Composite Noises (DataFrame)
def data_composite_noise (percent) :
    np.random.seed(3061996)
    x1 = np.random.uniform(-5, 5)
    x2 = np.random.uniform(-5, 5)
    a = np.linspace(min(x1,x2), max(x1, x2), 140)
    freq_power = 50
    freq_baseline = 0.333
    reduced_ratio_power = 1/4
    Vpp = 5
    power_noise = (np.sin(2 * np.pi * a *freq_power) * ( Vpp
*(reduced_ratio_power)*(percent/100)))
    v1 = np.random.uniform(0, 10)
    v2 = np.random.uniform(0, 10)
    b = np.linspace(min(v1,v2), max(v1, v2), 140)
    base_noise = (np.sin(2 * np.pi * b * freq_baseline) * ( Vpp *(percent/100)))
    mu = 0
    sigma = 1
    randon_num = np.random.normal(mu, sigma, 140)
    reduced_ratio_emg = 1/8
    emg_noise = randon_num*(Vpp)*(reduced_ratio_emg )*(percent/100)
    noise_combined = 0.5 *(power_noise + base_noise) + emg_noise
    data_comb = data_0_1.iloc[:,:-1] + noise_combined
    label = data_0_1.iloc[:,-1]
    data_comb = pd.concat( [data_comb, label ] ,axis =1 )
    return data_comb
# Call Function: "data_composite_noise"
# data_composite_noise (percent = 25)
```

```
## Define Function: "scatter_3d" to scatter 3D: window_size(w), word_size(p),
alphabet_size(a)
# Input: data
# Output: scatter 3D
def scatter_3d (data) :
    param_df = pd.DataFrame( data )
    fig = px.scatter_3d( param_df , x='w', y='p', z='a', color='accuracy',
color_continuous_scale=px.colors.sequential.Blues , color_continuous_midpoint =
0.5)
    # https://plotly.com/python/colorscales/
    fig['layout'].update(width=600, height=500, autosize=False)
    fig.show()
# Call Function: "scatter_3d"
# scatter_3d (data = data_0_1)


# Best parameter set with highest accuracy of Kfold Cross Validation in training data
(for each datasets)
# Return df: param, window, word, alphabet, best_accuracy_Kfoldtrainset
# Recheck: 1) All param Files in path 2) For each File contains columns: window,
word, alphabet, accuracy
csv_dict = {}
path =r' ## File path ## '
filename = [file for file in os.listdir(path) if file.startswith('param')]
for each_file in filename :
    file_noncsv = each_file.replace('.csv','')
    csv_dict[file_noncsv] = (pd.read_csv(path + (each_file)))
max_score_dict = {}
max_score_list = []
for i in csv_dict :
    max_score_dict [i] = csv_dict[i].sort_values(['accuracy'], ascending =
False)[:1].to_dict('records')
```

```
    max_score_df = csv_dict[i].sort_values(['accuracy'], ascending = False)[:1]
    max_score_df['param'] = i
    max_score_list.append(max_score_df.to_dict('records'))
max_score_df = pd.DataFrame(max_score_list)[0].apply(pd.Series).set_index('param')
max_score_df = max_score_df.rename(columns = { 'accuracy' :
'best_accuracy_Kfoldtrainset' }).reset_index()
max_score_df = max_score_df.sort_values('param', ascending = False)

## Define Function: "testset_model_score" to return  Accuracy in Testing Data for
SAXVSM or BOSSVS
# Input: data, model_str, window_size(w), word_size(p), alphabet_size(a)
# Output: Accuracy
def testset_model_score (data, model_str, w, p, a) :
    X = data.iloc[:,:-1].values
    y = data.iloc[:,-1].values
    X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.3 ,
random_state=1000)
    list_vs = []
    tfidf_dict_list = []
    model_str = model_str.lower()
    if model_str == 'sax' or model_str == 'saxvsm' :
        model =  SAXVSM ( window_size = w, word_size = p,  n_bins = a,
strategy='normal')
    elif model_str == 'boss' or model_str == 'bossvs' :
        model =  BOSSVS ( window_size = w, word_size = p,  n_bins = a,
strategy='quantile')
    try :
        model. fit ( X_train , y_train )
        tfidf = model.tfidf_
        keys = model.vocabulary_.values()
        tfidf_dict_list = [{key : val for key,  val in zip(keys, idx)} for idx in tfidf]
```

```
        list_vs.append(tfidf_dict_list)
        X_new = model.decision_function(X_test)
        y_pred = model.predict(X_test)
        vocabulary_length = len(model.vocabulary_)
        model_score_testset = model . score ( X_test , y_test )
    except UnboundLocalError :
        model_score_testset = 'Incorrect Model'
    return model_score_testset
# Call Function: "testset_model_score"
# SAX: Kfold_model_score( data = data_0_1 , model_str = 'SAX', w = 30, p =10, a =4
)
# BOSS: Kfold_model_score( data = data_0_1 , model_str = 'BOSS', w = 30, p =10, a
=4 )

## Code for Accuracy Precision Recall and F1 Score Comparison
test_time = {}
param_dict = {}
test_score = {}
# noise_pct = [ 25, 50, 100, 1000, 10000, 100000, 1000000, 10000000]
noise_pct = [ 25, 50, 100, 1000, 10000]
noise_type = [ 'emg', 'power', 'base', 'comb']
classify_algorithm = [ 'sax', 'boss']
for k in classify_algorithm :
    globals()[f'window_{k}'] = max_score_df ['window'] [max_score_df['param'] ==
'param_' + k ]
    globals()[f'word_{k}'] = max_score_df ['word'] [max_score_df['param'] == 'param_' +
k  ]
    globals()[f'alphabet_{k}'] = max_score_df ['alphabet'] [max_score_df['param'] ==
'param_' + k  ]
    param_dict['param_'+ k] = {'window': int(globals()[f'window_{k}']), 'word':
int(globals()[f'word_{k}']),
```

```
    'alphabet': int( globals()[f'alphabet_{k}'])}
sax_start = timeit.default_timer()
saxvsm_notnoise_acc, saxvsm_notnoise_precision, saxvsm_notnoise_recall,
saxvsm_notnoise_specificity, saxvsm_notnoise_f1_score  = testset_saxvsm_score
(data_0_1, int(window_sax), int(word_sax), int(alphabet_sax) )
sax_stop = timeit.default_timer()
sax_execution_time = sax_stop - sax_start
boss_start = timeit.default_timer()
bossvs_notnoise_acc , bossvs_notnoise_precision, bossvs_notnoise_recall,
bossvs_notnoise_specificity, bossvs_notnoise_f1_score = testset_bossvs_score
(data_0_1,int(window_boss),int(word_boss),int(alphabet_boss) )
boss_stop = timeit.default_timer()
boss_execution_time = boss_stop - boss_start
# test_score['data'] = { 'saxvsm' : saxvsm_notnoise_score_test, 'bossvs' :
bossvs_notnoise_score_test, 'saxvsm_precision' : saxvsm_notnoise_precision,
'bossvs_precision' : bossvs_notnoise_precision }
test_score['data'] = { 'saxvsm_acc' : saxvsm_notnoise_acc , 'saxvsm_precision' :
saxvsm_notnoise_precision ,
'saxvsm_recall' :saxvsm_notnoise_recall, 'saxvsm_specificity' :
saxvsm_notnoise_specificity, 'saxvsm_F1' :saxvsm_notnoise_f1_score ,
'time_sax' : sax_execution_time ,
'bossvs_acc' : bossvs_notnoise_acc, 'bossvs_precision' : bossvs_notnoise_precision ,
'bossvs_recall' : bossvs_notnoise_recall, 'bossvs_specificity' :
bossvs_notnoise_specificity, 'bossvs_F1' : bossvs_notnoise_f1_score ,
'time_boss' : boss_execution_time }
for i in noise_type :
    for j in noise_pct :
        for k in classify_algorithm :
            globals()[f'window_{k}_{i}{str(j)}'] = max_score_df ['window']
[max_score_df['param'] == 'param_' + k + '_' + i + str(j) ]
```

```
            globals()[f'word_{k}_{i}{str(j)}'] = max_score_df ['word'] [max_score_df['param']
== 'param_' + k + '_' + i + str(j) ]
            globals()[f'alphabet_{k}_{i}{str(j)}'] = max_score_df ['alphabet']
[max_score_df['param'] == 'param_' + k + '_' + i + str(j) ]
            window = globals()[f'window_{k}_{i}{str(j)}']
            word = globals()[f'word_{k}_{i}{str(j)}']
            alphabet = globals()[f'alphabet_{k}_{i}{str(j)}']
            param_dict[ 'param_' + k + '_' + i + str(j) ] = {'window' : int(window) , 'word' :
int(word) , 'alphabet' : int(alphabet)}
            #print(param_dict[ 'param_' + k + '_' + i + str(j) ])
        globals()[f'data_{i}{str(j)}'] = eval( 'data_' + i + '_noise (' + str(j) +')' )
        sax_start = timeit.default_timer()
        saxvsm_acc, saxvsm_precision, saxvsm_recall, saxvsm_specificity,
saxvsm_f1_score  = testset_saxvsm_score(globals()[f'data_{i}{str(j)}'],
int(globals()[f'window_sax_{i}{str(j)}']),int(globals()[f'word_sax_{i}{str(j)}']) ,
int(globals()[f'alphabet_sax_{i}{str(j)}'] ) )
        sax_stop = timeit.default_timer()
        sax_execution_time = sax_stop - sax_start
        #print("Program Executed in "+str(sax_execution_time))
        boss_start = timeit.default_timer()

        bossvs_acc , bossvs_precision, bossvs_recall, bossvs_specificity, bossvs_f1_score
= testset_bossvs_score(globals()[f'data_{i}{str(j)}'],
int(globals()[f'window_boss_{i}{str(j)}']),int(globals()[f'word_boss_{i}{str(j)}']) ,
int(globals()[f'alphabet_boss_{i}{str(j)}'] ) )
        boss_stop = timeit.default_timer()
        boss_execution_time = boss_stop - boss_start
        #print("Program Executed in "+str(boss_execution_time))
        test_score['data_' + i + '_noise' + str(j) ] = { 'saxvsm_acc' : saxvsm_acc ,
'saxvsm_precision' : saxvsm_precision ,
```

```
            'saxvsm_recall' :saxvsm_recall, 'saxvsm_specificity' : saxvsm_specificity,
'saxvsm_F1' :saxvsm_f1_score ,'time_sax' : sax_execution_time  ,
            'bossvs_acc' : bossvs_acc, 'bossvs_precision' : bossvs_precision ,
            'bossvs_recall' : bossvs_recall, 'bossvs_specificity' : bossvs_specificity,
'bossvs_F1' : bossvs_f1_score ,
            'time_boss' : boss_execution_time }
test_score_df = pd.DataFrame(test_score).T.reset_index()
test_score_df[['data','noise_type', 'pct']] = test_score_df['index'].str.split('_',
expand=True)
test_score_df['pct'] = test_score_df['pct'].str.replace('noise','')
test_score_df = test_score_df.drop(columns= 'data')
test_score_df = test_score_df.rename(columns = {'index' : 'data'})
test_score_df['pct'] = test_score_df['pct'].fillna(0)
test_score_df['noise_type'] = test_score_df['noise_type'].fillna('None')
test_score_df['pct'] = test_score_df['pct'].astype(str)
test_score_df['noise_type'].replace({'power', 'powerline'})
test_score_df['noise_type'] = test_score_df['noise_type'].replace(  {'power' :
'powerline' , 'base' : 'baseline', 'comb' : 'composite' } )

## Code for Run time calculation
start = timeit.default_timer()
   ## Code Statement ##
stop = timeit.default_timer()
execution_time = stop - start
print("Program Executed in "+str(execution_time))
time_dict["param_boss_comb10000"] = execution_time
```

ประวัติผู้เขียน

ชื่อ-สกุล นางสาวนภัสสร แก้วกล้า

วัน เดือน ปี เกิด 3 มิถุนายน 2539

วุฒิการศึกษา วิทยาศาสตรบัณฑิต สาขาคณิตศาสตร์ประยุกต์ มหาวิทยาลัยธรรมศาสตร์