ระบบระบุตำแหน่งภายในอาคาร โดยใช้วิธีการเรียนรู้เครื่องจักรร่วมกับอัลกอริทึม Kalman Filter

ธนสิทธิ์ ฤทธิ์ธนโสภณ

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
บัณฑิตวิทยาลัย สาขาเทคโนโลยีวิศวกรรม
สถาบันเทคโนโลยีไทย-ญี่ปุ่น
ปีการศึกษา 2565

INTEGRATION OF MACHINE LEARNING AND KALMAN FILTER APPROACH FOR FINGERPRINT INDOOR POSITIONING

Thanasit Rithanasophon

A Thesis Submitted in Partial Fulfillment of Requirements
for The Degree of Master of Engineering Program in Engineering Technology
Graduate School
Thai-Nichi Institute of Technology
Academic Year 2022

หัวข้อวิทยานิพนธ์	ระบบระบุตำแหน่งภายในอาคาร โดยใช้วิธีการเรียนรู้เครื่องจักรร่วมกับอัลกอริทึม Kalman Filter
โดย	ธนสิทธิ์ ฤทธิ์ธนโสภณ
สาขาวิชา	เทคโนโลยีวิศวกรรม
อาจารย์ที่ปรึกษาวิทยานิพนธ์	ดร.ชัชไชย วรรณบูรณ์

บัณฑิตวิทยาลัย สถาบันเทคโนโลยีไทย-ญี่ปุ่น อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

.....คณบดีบัณฑิตวิทยาลัย
(รองศาสตราจารย์ ดร.วรากร ศรีเชวงทรัพย์)
วันที่.....เดือน.....พ.ศ.....

คณะกรรมการสอบวิทยานิพนธ์

.....ประธานกรรมการ
(ดร.ณัฐกิตติ์ จิตรเอื้อตระกูล)

.....กรรมการ
(รองศาสตราจารย์ ดร.วรากร ศรีเชวงทรัพย์)

.....กรรมการ
(ดร.อัดนา เซนโต๊ะ)

.....อาจารย์ที่ปรึกษาวิทยานิพนธ์
(ดร.ชัชไชย วรรณบูรณ์)

ธนสิทธิ์ ฤทธิ์ธนโสภณ : ระบบระบุตำแหน่งภายในอาคาร โดยใช้วิธีการเรียนรู้เครื่องจักรร่วมกับอัลกอริทึม Kalman Filter. อาจารย์ที่ปรึกษา : ดร.ชัชไชย วรรณบูรณ์, 50 หน้า.

ระบบระบุตำแหน่งภายในอาคารด้วยวิธีการ Fingerprint นั้นเป็นระบบที่เรียบง่ายและถูกใช้อย่างแพร่หลายในการระบุตำแหน่งของอุปกรณ์ต่างๆ ภายในอาคารหรือพื้นที่ปิด แต่เนื่องจากความแปรปรวนของสภาพแวดล้อม ทำให้สัญญาณที่ตรวจวัดได้มีความไม่แน่นอน และเกิดความคาดเคลื่อนของตำแหน่งสูง ในงานวิจัยนี้ได้นำเสนอวิธีการการเรียนรู้ของเครื่องจักร (Machine Learning) ร่วมกับอัลกอริทึม Kalman Filter เพื่อเพิ่มความแม่นยำของระบบระบุตำแหน่งด้วยวิธีการ Fingerprint ซึ่งวิธีการที่นำเสนอนั้นได้นำความสามารถของการเรียนรู้ของเครื่องจักรมาใช้ในการถอดคุณสมบัติและการจำแนกประเภท รวมถึงนำความสามารถในการกำจัดค่าความแปรปรวนของสัญญาณด้วย Kalman Filter มาผนวกเข้าด้วยกัน การทดลองนี้ได้ทำการรวบรวมชุดข้อมูลจากสภาพแวดล้อมจริงที่ได้ทำการเก็บมาจากอุปกรณ์กระจายสัญญาที่ใช้เทคโนโลยีบลูทูธพลังงานต่ำที่ถูกวางไว้ในจุดต่างๆ ภายในพื้นที่การทดลอง ผลลัพธ์จากการทดสอบระบบ พบว่าวิธีการที่นำเสนอนั้นสามารถเพิ่มค่าความแม่นยำของระบบระบุตำแหน่งภายในอาคารด้วยวิธีการ Fingerprint ได้ โดยเปรียบเทียบกับวิธีการเรียนรู้ด้วยเครื่องแบบดั้งเดิม ซึ่งเป็นต้นแบบของระบบระบุตำแหน่งภายในอาคาร ที่มีต้นทุนต่ำแต่ได้ค่าประสิทธิภาพและความแม่นยำที่สูง

บัณฑิตวิทยาลัย ลายมือชื่อนักศึกษา .....
สาขาวิชา เทคโนโลยีวิศวกรรม ลายมือชื่ออาจารย์ที่ปรึกษา .....
ปีการศึกษา 2565

THANASIT RITHANASOPHON : INTEGRATION OF MACHINE LEARNING AND KALMAN FILTER APPROACH FOR FINGERPRINT INDOOR POSITIONING. ADVISOR : DR.CHATCHAI WANNABOON, 50 PP.

Fingerprint-based indoor positioning systems are simple and widely used to determine the location of a device inside a building or other enclosed area. However, the accuracy and reliability are still a major concern due to the turbulence in the environment and the presence of noise in the data. This paper presents a machine learning integrated with Kalman filter approach for improving the accuracy of fingerprint-based indoor positioning systems. The proposed approach combines the power of machine learning techniques for feature extraction and classification with the noise-filtering capabilities of the Kalman filter. Implementation is achieved by a real-world dataset collected from multiple Bluetooth low energy access points. The experiment results indicate that the proposed approach significantly improves the accuracy of fingerprint-based indoor positioning compared to traditional machine learning approaches. This study also offers a potential of cost-effective and high accuracy algorithm for the indoor positioning applications.

Graduate School
Field of Engineering of Technology
Academic Year 2022

Student's Signature.....
Advisor's Signature.....

กิตติกรรมประกาศ

งานวิทยานิพนธ์ฉบับนี้ทำวิจัยเกี่ยวกับเรื่อง “ระบบระบุตำแหน่งภายในอาคาร โดยใช้วิธีการเรียนรู้เครื่องจักรร่วมกับอัลกอริทึม Kalman Filter” สำเร็จลุล่วงไปได้ด้วยความกรุณาอย่างยิ่งจาก ดร.ชัชไชย วรรณบูรณ์ อาจารย์ที่ปรึกษาซึ่งได้ให้ความรู้ คำปรึกษา แนะนำแนวทาง เพื่อนำไปปรับปรุง และแก้ไขข้อบกพร่องต่างๆ เพื่อให้งานสมบูรณ์ยิ่งขึ้น พร้อมทั้งยังช่วยพัฒนาการทำงานของผู้วิทยานิพนธ์ให้เป็นไปอย่างมีคุณภาพ และสนับสนุนเครื่องมือที่ใช้ในการทำงานวิจัยนี้ อีกทั้งยังดูและเอาใจใส่ดีตลอดระยะเวลาในการทำวิทยานิพนธ์

ขอขอบพระคุณคณะกรรมการทุกท่านที่ได้ให้ความรู้ คำแนะนำ และข้อคิดที่เป็นประโยชน์ต่อการทำวิทยานิพนธ์ฉบับนี้ พร้อมทั้งให้คำแนะนำแนวทาง ปรับปรุงแก้ไข จนสามารถทำให้วิทยานิพนธ์ฉบับนี้เสร็จสิ้นสมบูรณ์ตามขอบเขตที่กำหนดไว้

นอกจากนี้ทางผู้วิจัยขอขอบคุณแรงสนับสนุนจากครอบครัว และคนรอบข้างที่คอยให้กำลังใจให้คำปรึกษาตลอดระยะเวลาการศึกษา จนจัดทำวิทยานิพนธ์ฉบับนี้เสร็จสิ้น

ผู้วิจัยหวังเป็นอย่างยิ่งว่า วิทยานิพนธ์ฉบับนี้จะก่อให้เกิดประโยชน์แก่ผู้ที่สนใจ และสามารถนำไปต่อยอดกับองค์ความรู้ที่เกี่ยวข้องได้ อีกทั้งสามารถก่อให้เกิดแนวคิดเพื่อใช้ในการศึกษา หรือเป็นแนวทางการทำงานวิจัยที่เกี่ยวข้องในอนาคต

ธนสิทธิ์ ฤทธิ์ธนโสภณ

สารบัญ

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญรูป

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

อินเตอร์เน็ตในทุกสิ่ง (Internet of Things: IoT) เป็นสิ่งที่กำลัง ได้รับความสนใจอย่างกว้างขวาง เนื่องจากเป็นเทคโนโลยีที่ส่งผลให้เกิดสภาพแวดล้อมทางด้านนวัตกรรม (Innovation Ecosystem) โดยที่อุปกรณ์ในสมัยใหม่นั้นสามารถที่จะเชื่อมต่อกันผ่านเครือข่ายอินเตอร์เน็ตทำให้อุปกรณ์ต่างๆ สามารถสื่อสารและถูกควบคุมผ่านเครือข่ายอินเตอร์เน็ตได้ก่อให้เกิดเทคโนโลยีการควบคุมแบบไร้สาย อีกทั้งยังสามารถติดตามอุปกรณ์ต่างๆ ผ่านระบบอินเตอร์เน็ตได้อีกด้วย ทำให้เกิดระบบที่นำเทคโนโลยีอินเตอร์เน็ตในทุกสิ่งไปประยุกต์ใช้ อาทิเช่น ระบบควบคุมแสงไฟภายในบ้าน ระบบปรับความเย็นอัจฉริยะ ระบบควบคุมประตูอัจฉริยะ เป็นต้น

นอกจากการนำเทคโนโลยีอินเตอร์เน็ตในทุกสิ่งไปประยุกต์ใช้กับระบบอัจฉริยะแล้ว เทคโนโลยีนั้นยังมีการนำไปประยุกต์ใช้กับระบบระบุพิกัดอีกด้วย ซึ่งระบบระบุพิกัดนั้นสามารถที่จะช่วยในการติดตามสิ่งของ ติดตามยานพาหนะ ติดตามสัตว์เลี้ยง หรือแม้แต่ติดตามบุคคลอีกด้วย ซึ่งมีประโยชน์ในหลายๆ ด้าน โดยที่สามารถติดตามหรือเก็บข้อมูลพิกัดของสิ่งนั้นเพื่อใช้ในการตามหาในกรณีสูญหาย ซึ่งนำข้อมูลพิกัดที่ได้ไปวิเคราะห์เพื่อทำการกู้คืนของสิ่งนั้น หรือสามารถติดตามข้อมูลตำแหน่งพิกัดเพื่อใช้ในการวิเคราะห์ในเชิงพาณิชย์ได้อีกด้วย ทำให้งานวิจัยทางด้านระบบระบุพิกัดมีความน่าสนใจ และสามารถนำไปใช้ประโยชน์ได้อย่างหลากหลาย อีกทั้งระบบติดตามหรือระบุพิกัดนั้นมีทั้งภายนอกอาคารและภายในอาคาร โดยที่เทคโนโลยีที่ใช้นั้นมีความแตกต่างกัน

ระบบระบุพิกัดนั้นมี 2 ประเภทคือ ระบบระบุพิกัดภายนอกอาคาร และระบบระบุพิกัดภายในอาคาร โดยที่ระบบระบุพิกัดภายนอกอาคารนั้นมีเทคโนโลยีหลักที่ใช้ในการระบุตำแหน่งคือ Global Navigation Satellite Systems (GNSS) ซึ่งใช้ดาวเทียมในการระบุพิกัดโดยที่มีระบบที่รู้จักกันอย่างกว้างขวาง อาทิเช่น GPS ซึ่งสามารถใช้ระบุพิกัดได้อย่างแม่นยำ โดยมีความคลาดเคลื่อนน้อยกว่า 5 เมตร [1] ในที่โล่งแจ้ง แต่เทคโนโลยีนี้ก็มีข้อเสียเช่นกัน กล่าวคือ ความคลาดเคลื่อนจะเพิ่มมากขึ้นเมื่ออยู่ในพื้นที่ปิดหรือตัวอาคาร ทำให้ไม่สามารถใช้เทคโนโลยี GPS ในการระบุตำแหน่งภายในอาคารได้

ดังนั้น งานวิจัยนี้จึงมุ่งเน้นในการวิจัยเพื่อเพิ่มประสิทธิภาพของระบบระบุพิกัดภายในอาคาร ด้วย วิธีการใช้โทรศัพท์มือถือกับสัญญาณบลูทูธพลังงานต่ำ (BLE) เนื่องจากบลูทูธนั้นมีข้อดีในด้านการใช้พลังงานที่ต่ำและง่ายต่อการใช้งาน โดยจะทำการระบุตำแหน่งพิกัดโดยการวัดความแรงของสัญญาณบลูทูธของอุปกรณ์ระบบฝังตัวกับเสาสัญญาณ ซึ่งวัดจากตัวบ่งชี้ความแรงของสัญญาณ (RSSI) ของอุปกรณ์ระบบฝังตัวกับเสาสัญญาณเพื่อใช้ในการวิเคราะห์และทำนายความห่างของอุปกรณ์ระบบฝังตัวกับเสารับ

สัญญาณ ณ จุดต่างๆ แต่เนื่องจากสัญญาณที่ได้รับนั้นมีความไม่คงที่ ดังนั้นจึงได้นำอัลกอริทึม Kalman Filter ที่ใช้ในการคำนวณตำแหน่งของวัตถุจากการเคลื่อนที่ก่อนหน้าเพื่อใช้ในการกรองค่าของตัวบ่งชี้ความแรงของสัญญาณ อีกทั้งยังได้นำเทคโนโลยีการเรียนรู้เครื่องจักร (Machine Learning) มาใช้ในการฝึกสอนข้อมูลที่ถูกเก็บรวบรวมมาเพื่อทำให้การระบุตำแหน่งมีความแม่นยำมากยิ่งขึ้นและสามารถนาไปใช้ได้จริงกับการระบุพิกัดภายในอาคาร

1.2 วัตถุประสงค์ของการศึกษา

เพื่อเพิ่มประสิทธิภาพให้กับระบบระบุพิกัดภายในอาคารให้มีความแม่นยำมากขึ้นและสามารถนำไปใช้ได้จริง

1.3 ขอบเขตของการศึกษา

1.3.1 เก็บข้อมูลความแรงของสัญญาณบลูทูธจากตัวกระจายสัญญาณ Ruuvi และรับด้วยอุปกรณ์ฝังตัว Raspberry Pi 4 ด้วยเทคโนโลยีบลูทูธพลังงานต่ำ

1.3.2 นำข้อมูลที่ได้ไปใช้ในการฝึกสอนโมเดลด้วยวิธีการเรียนรู้ด้วยเครื่อง

1.3.3 ประเมินวัดผลโมเดลที่ได้จากการฝึกสอน

1.4 ขั้นตอนการดำเนินงาน

1.4.1 ออกแบบพื้นการทดลอง และกำหนดตำแหน่งจุดวางตัวกระจายสัญญาณ Ruuvi

1.4.2 เก็บรวบรวมข้อมูลค่าความแรงของสัญญาณด้วย Raspberry Pi 4 ณ จุดต่างๆ ของพื้นที่การทดลอง

1.4.3 คัดเลือกอัลกอริทึมที่ใช้ในการฝึกสอนโมเดลและเปรียบเทียบผลลัพธ์

1.4.4 ประเมินวัดผลความแม่นยำของโมเดล

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1.5.1 สามารถเก็บข้อมูลค่าความแรงของสัญญาณได้

1.5.2 สามารถนำข้อมูลที่ถูกเก็บรวบรวมไปใช้ในการฝึกสอนโมเดลด้วยวิธีการเรียนรู้ด้วยเครื่องได้

1.5.3 สามารถประเมินวัดผลและเปรียบเทียบความแม่นยำของโมเดลได้

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

ในบทนี้จะนำเสนอทฤษฎีที่เกี่ยวข้องกับระบบระบุตำแหน่งภายในอาคาร โดยใช้วิธีการเรียนรู้เครื่องจักรร่วมกับอัลกอริทึม Kalman Filter และการทบทวนวรรณกรรมผู้วิจัยได้ทำการศึกษาเอกสารทฤษฎี และงานวิจัยต่างๆ ที่เกี่ยวข้อง โดยทฤษฎีที่เกี่ยวข้องมีดังต่อไปนี้

2.1 ทฤษฎีที่เกี่ยวข้อง

2.1.1 ระบบระบุพิกัดภายในอาคาร (Indoor Positioning System : IPS)

ระบบระบุพิกัดภายในอาคาร คือ ระบบที่ใช้ในการระบุตำแหน่งหรือพิกัดของอุปกรณ์ที่สามารถส่งสัญญาณคลื่นวิทยุได้โดยที่อุปกรณ์จะอยู่ภายในอาคารหรือพื้นที่ปิด ซึ่งเทคโนโลยีที่นำมาใช้งานมีหลากหลายชนิด โดยเทคโนโลยีที่ใช้ในการวัดระยะของอุปกรณ์นั้นได้แก่ ตัวบ่งชี้ความแรงของสัญญาณ (Received-Signal-Strength-Indicator : RSSI), ระยะเวลาในการมาถึง (Time-of-Arrival : ToA), ความต่างของระยะเวลาในการมาถึง (Time-Difference-of-Arrival : TDoA) และองศาในการมาถึง (Angle-of-Arrival : AoA) อีกทั้งยังมีเทคโนโลยีที่ใช้ในการสื่อสารระหว่างอุปกรณ์ส่งสัญญาณและรับสัญญาณก็มีหลากหลาย ได้แก่ Wi-Fi, Bluetooth, Ultra wide Band (UWB) และ Radio-Frequency Identification Tags (RFID)

โดยที่ระบบระบุพิกัดภายในอาคารนั้นจำเป็นต้องมีอุปกรณ์อย่างน้อย 2 ตัวที่ถูกติดตั้งอยู่ในพื้นที่เพื่อที่จะใช้ในการตรวจจับอุปกรณ์และคำนวณระยะทางระหว่างอุปกรณ์ซึ่งใช้ในการระบุตำแหน่งของอุปกรณ์ที่สนใจ วิธีการที่นำมาใช้ในการคำนวณนั้นได้แก่ วิธีการ Multilateration และ Trilateration ซึ่งวิธีการนี้จะทำการวัดระยะของอุปกรณ์ที่ถูกติดตั้งกับอุปกรณ์ที่สนใจ อุปกรณ์ที่ถูกติดตั้งจำเป็นต้องมีอย่างน้อย 3 ตัวขึ้นไป เมื่อได้ระยะของอุปกรณ์แล้วจะทำการวาดวงกลมโดยที่จุดตัดของวงกลมทั้ง 3 จุดจะกลายเป็นตำแหน่งของอุปกรณ์ที่คำนวณออกมาได้ วิธีการ Triangulation วิธีการนี้จะวัดโดยใช้องศาในการมาถึง (AoA) โดยที่อุปกรณ์ที่ถูกติดตั้งต้องใช้อย่างน้อย 2 ตัว อีกทั้งความแม่นยำของวิธีการนี้จะขึ้นอยู่กับจำนวนของอุปกรณ์ที่ถูกติดตั้ง ซึ่งจำนวนที่มากจะส่งผลต่อความแม่นยำที่สูง และวิธีการ Fingerprinting โดยวิธีการนี้จะทำการแบ่งออกเป็น 2 ขั้นตอน คือ ขั้นตอน Offline และขั้นตอน Online ซึ่งส่วนของขั้นตอน Offline จะเป็นส่วนของการเก็บข้อมูลสัญญาณแล้วนำข้อมูลที่ได้มาจัดกลุ่ม และในส่วนของขั้นตอน Online จะทำการวัดค่าสัญญาณจริงมาเปรียบเทียบกับกลุ่มสัญญาณที่อยู่ในขั้นตอน Offline เพื่อที่จะได้คำตอบเป็นจุดของตำแหน่งนั้นๆ [2]

2.1.2 ตัวบ่งชี้ความแรงของสัญญาณ (Received Signal Strength Indicator : RSSI)

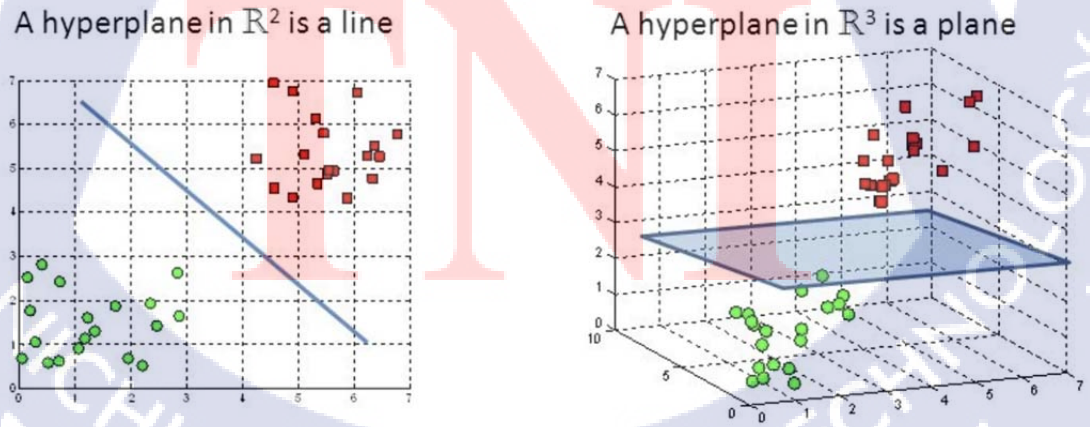
ตัวบ่งชี้ความแรงของสัญญาณ คือ มาตราวัดของค่าความแรงของสัญญาณที่ระบุว่าเครื่องมือวัด สัญญาณนั้นสามารถรับสัญญาณจากจุด Access Point หรือ Router ได้ดีแค่ไหน ซึ่งก็คือค่าที่ใช้ในการบอกคุณภาพของสัญญาณที่ได้รับจากมาตรฐาน IEEE802.11 ระบุว่า ค่า RSSI มาตราวัดนั้นสามารถมีค่าได้ตั้งแต่ 0-255 ซึ่งขึ้นอยู่กับผู้ผลิตอาทิเช่น Cisco ใช้มาตราวัดที่ 0-100 หรือAtherosใช้มาตราวัดที่ 0-60 โดยที่ยิ่งค่า RSSI ยิ่งสูงหมายความว่าความแรงของสัญญาณยิ่งดี [3]

2.1.3 เทคโนโลยีบลูทูธพลังงานต่ำ (Bluetooth Low Energy : BLE)

บลูทูธพลังงานต่ำเป็นอุปกรณ์ที่ถูกใช้กับอุปกรณ์ต่างๆ มากที่สุดเมื่อเทียบกับสัญญาณวิทยุอื่นๆ ซึ่งอยู่ในมาตรฐาน IEEE 802.15 โดยที่สัญญาณบลูทูธเป็นคลื่นแม่เหล็กไฟฟ้าที่ทำอยู่ในช่วงคลื่นความถี่ 2.4 GHz ถึง 2.4835 GHz ในปี 2013 บริษัท Apple ได้ผลิต iBeacon ซึ่งถูกพัฒนามาจากสัญญาณบลูทูธพลังงานต่ำ โดยที่ iBeacon นั้นสามารถส่งข้อมูลได้จากโทรศัพท์มือถือโดยตรง อีกทั้งยังใช้พลังงานต่ำและราคาต่ำกว่าบลูทูธปกติกับเทคโนโลยี Wi-Fi อีกด้วย [4]

2.1.4 Support Vector Machine

Support Vector Machine (SVM) [5] คือ อัลกอริทึมที่ใช้ในการหา Hyperplane หรือเส้นที่ลากผ่านเพื่อใช้ในการแบ่งประเภทของ Data Point โดย Hyperplane จะได้จากการคำนวณหา Margin ที่มีค่าระยะห่างจากจุด Data Point มากที่สุดของแต่ละ Class ซึ่งชุดข้อมูลนั้นสามารถมีได้ตั้งแต่ 2 Classes ขึ้นไป อีกทั้งการแยกประเภทนั้นจะใช้วิธีการแบ่งฝั่งจาก Hyperplane ซึ่งหมายความว่าแต่ละฝั่งของ Hyperplane จะเท่ากับ 1 Class



รูปที่ 2.1 ตัวอย่างลักษณะของ Hyperplane แบบ 2 และ 3 คุณลักษณะ

ค่า Margin ของ SVM นั้นจะขึ้นอยู่กับความแตกต่างของคุณลักษณะของแต่ละ Data Point ยิ่งแตกต่างมากเท่าไหร่ก็จะทำให้ค่า Margin มีค่ามากตามไปด้วย โดยงานวิจัยนี้ได้ใช้ Radial Basis Function (RBF) [6] เพื่อใช้ในการคำนวณหา Margin ของ Data Point สมการของ RBF เป็นดังนี้

$$K(X_1, X_2) = \exp\left(-\frac{\|X_1 - X_2\|^2}{2\sigma^2}\right) \quad (2.1)$$

โดยที่

σ คือ ค่าความแปรปรวนของ Hyperparameter

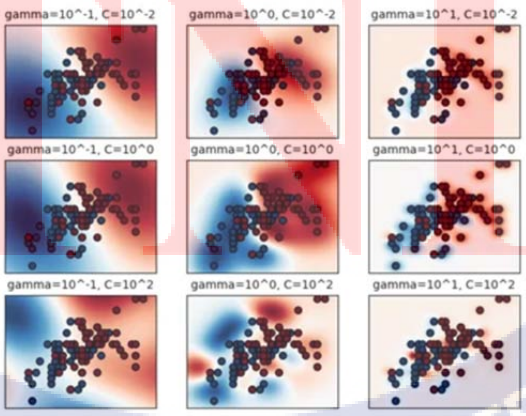
$\|X_1 - X_2\|$ คือ ค่า Euclidean Distance ของจุดที่ 1 และ 2

ดังนั้นสมการสามารถเขียนใหม่เป็น

$$K(X_1, X_2) = \exp\left(-\frac{d_{12}}{2\sigma^2}\right) \quad (2.2)$$

ซึ่งค่าสูงที่เป็นไปได้ของผลลัพธ์จากสมการนี้จะเท่ากับ 1 และจะเกิดขึ้นเมื่อ d_{12} มีค่าเท่ากับ 0 ซึ่งหมายความว่าทั้ง 2 จุดคือจุดเดียวกัน อีกทั้งความสำคัญของการใช้อัลกอริทึมนี้คือการปรับค่า σ เพื่อให้เหมาะสมกับชุดข้อมูล

โดยอัลกอริทึม SVC ใน Library Scikit-Learn จะมี Hyperparameter 2 ค่า คือ C และ $\gamma\left(\alpha\frac{1}{\sigma}\right)$ ซึ่งตัวอย่างของการปรับค่าเป็นดังนี้

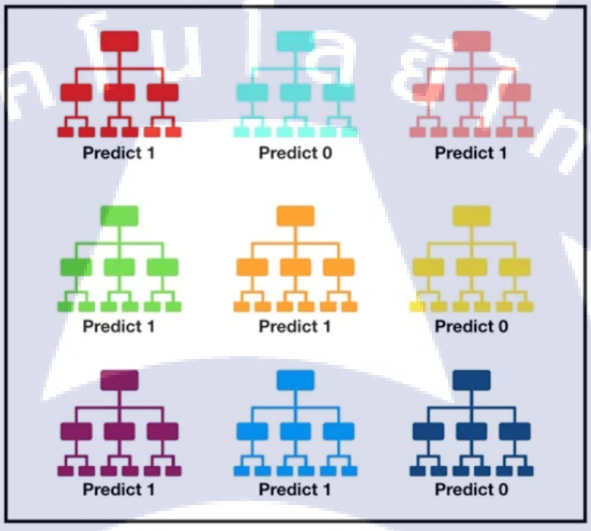


รูปที่ 2.2 ตัวอย่างผลลัพธ์ของการปรับค่า Hyperparameter C และ γ

ซึ่งจะเห็นว่าการเพิ่มค่า γ จะทำให้แนวโน้มของโมเดลมีความ Overfit มากขึ้นตามค่า C ที่กำหนดไว้ในแต่ละครั้ง

2.1.5 Random Forest

Random Forest [7] คือ Ensemble Model ชนิดหนึ่งที่ประกอบไปด้วย Decision Tree จำนวนมากที่ไม่สัมพันธ์กันมาทำการทำนายประเภทและทำการ Voting โดยการพิจารณาว่าคำตอบไหนที่ถูกทำนายมากที่สุดก็จะถูกเลือกเป็นคำตอบของโมเดล



รูปที่ 2.3 ตัวอย่างผลลัพธ์ที่ได้จากการ Voting ของ Random Forest

โดย Decision Tree คือ อัลกอริทึมประเภท Supervised Machine Learning ที่ประกอบไปด้วย

- Root Node คือ จุดกำเนิดของ Decision Tree
- Internal Node คือ จุดที่ใช้ในการประเมินตามเงื่อนไข
- Leaf Node คือ จุดที่ใช้ในการตัดสินใจ

ซึ่งในงานวิจัยนี้ได้เลือกใช้วิธีการ Gini Impurity ซึ่ง Gini Impurity คือฟังก์ชันที่ใช้วัดความสามารถในการแตกกิ่งของ Decision Tree และมีค่าอยู่ในช่วง 0 ถึง 0.5 สมการของ Gini Impurity จะแสดงได้ดังนี้

$$Gint(t) = 1 - \sum_{i=1}^{j} P(i \mid t)^2 \quad (2.3)$$

โดยที่

j คือ จำนวน Class ของชุดข้อมูล

P คือ Ratio ของ Class ที่ Node ณ ตำแหน่งต่างๆ

ผลลัพธ์จากสมการนี้ยิ่งมีค่าเข้าใกล้ 0 เท่าไหร่จะหมายความว่ามีความสามารถในการแตกกิ่งที่ดี [8]

2.1.6 Gaussian Naïve Bayes

Gaussian Naïve Bayes [9] คือ ส่วนขยายของ Naïve Bayes โดยที่ Naïve Bayes คือ อัลกอริทึม Machine Learning ประเภท Probabilistic โดยทฤษฎี Bayes มีสมการดังนี้

$$P(Y|X) = P(X|Y) \times \frac{P(Y)}{P(X)} \quad (2.4)$$

โดยที่

P(Y) คือ ความน่าจะเป็นของการเกิดเหตุการณ์ A (Class Prior Probability)

P(X) คือ ความน่าจะเป็นของการเกิดเหตุการณ์ B (Predictor Prior Probability)

P(Y|X) คือ ความน่าจะเป็นของเหตุการณ์ A ภายในเหตุการณ์ B (Posterior Probability)

P(X|Y) คือ ความน่าจะเป็นของเหตุการณ์ B ภายในเหตุการณ์ A (Likelihood)

Gaussian Naïve Bayes นั้นใช้ร่วมกับข้อมูลรูปแบบเชิงปริมาณที่มีความต่อเนื่อง (Continuous Data) ซึ่งทำให้ข้อมูลคุณลักษณะอยู่ในรูปแบบ Normal (หรือ Gaussian) Distribution และแต่ละคุณลักษณะไม่มีความสัมพันธ์ต่อกัน ดังนั้นสมการจะแตกต่างจาก Naïve Bayes โดยการใช้สมการหา Likelihood โดยสมการเป็นดังนี้

$$P(X|Y = c) = \frac{1}{\sqrt{2\pi\sigma_c^2}} e^{\frac{-(x-\mu_c)^2}{2\sigma_c^2}} \quad (2.5)$$

โดยที่

μ_c คือ ค่ากลางของ Class นั้นๆ

σ_c คือ ค่าความแปรปรวนของ Class นั้นๆ

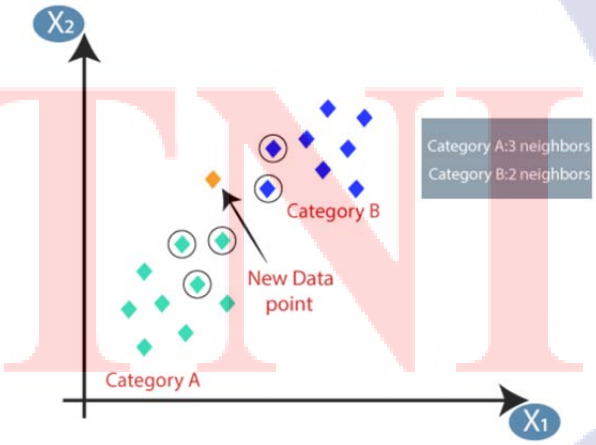
ผลลัพธ์ที่ได้จากการคำนวณจะเป็นค่าความน่าจะเป็นของ Class นั้นๆ ซึ่งค่าความน่าจะเป็นที่มีค่ามากที่สุดก็จะเป็นคำตอบว่าข้อมูลชุดนั้นตกอยู่ใน Class ใด

2.1.7 K-nearest Neighbors

K-nearest Neighbors (KNN) [10] คือ อัลกอริทึมที่ใช้ในการหาความใกล้เคียงของข้อมูลจากชุดข้อมูลที่ใช้ในการฝึกสอนเพื่อจัดกลุ่มให้กับ class โดยทำการเลือกจำนวนชุดข้อมูลที่ใกล้ที่สุดจำนวน K ตัว โดยคำนวณหาระยะทางของข้อมูลด้วยสมการ Euclidean Distance ซึ่งสมการเป็นดังนี้

$$\text{Euclidean Distance} = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2} \qquad (2.6)$$

เมื่อทำการหาระยะทางของข้อมูลทั้งหมด K ตัวแล้วจึงนำมาตรวจสอบว่าข้อมูลทั้ง K ตัวนั้นอยู่ใน Class ใดบ้าง โดยที่จำนวน Class ที่มากที่สุดเป็น Class ใดก็จะนับว่าข้อมูลชุดนั้นอยู่ใน Class นั้น เช่นกัน



รูปที่ 2.4 ตัวอย่างผลลัพธ์จากการคำนวณหา Class ที่ใกล้ที่สุดของชุดข้อมูลที่มีจำนวน K เท่ากับ 5 และจากผลลัพธ์จะจัดกลุ่มข้อมูลใหม่ให้อยู่ใน Category A

2.1.8 Kalman Filter

Kalman Filter คือ อัลกอริทึมที่ใช้ในการคาดคะเนตำแหน่งถัดไปของวัตถุจากข้อมูลก่อนหน้า ซึ่ง Kalman Filter ที่ใช้อยู่ในรูปแบบการทำนายวัตถุที่ไม่เคลื่อนที่ ดังนั้น Transition Model และ Observation Model จะเท่ากับ

$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t = x_{t-1} + \varepsilon_t \quad (2.7)$$

$$z_t = C_t x_t + \delta_t = x_t + \delta_t \quad (2.8)$$

โดยที่ ε_t และ δ_t คือ Gaussian Noise ในส่วนของตัวแปร A_t, B_t และ C_t คือ Transition Model อีกทั้ง A_t และ C_t ถูกแทนค่าด้วย Identity Matrices (I_n) และ Control Model ถูกแทนค่าด้วย 0 ทำให้ Estimate State และ Covariance มีสมการดังนี้

$$\bar{u}_t = u_{t-1} \quad (2.9)$$

$$\overline{\sum}_t = \sum_{t-1} + R_t \quad (2.10)$$

โดยที่ R_t คือ Process Noise และสมการ Kalman Gain จะเป็นดังนี้

$$K_t = \overline{\sum}_t (\overline{\sum}_t + Q_t)^{-1} \quad (2.11)$$

โดยที่ Q_t คือ Observation Noise ซึ่งมีค่าที่ขึ้นอยู่กับเครื่องมือวัด อีกทั้ง State Estimate และ Estimate Covariance ของเครื่องมือวัดใหม่ที่ถูกแก้ไขเป็นดังนี้

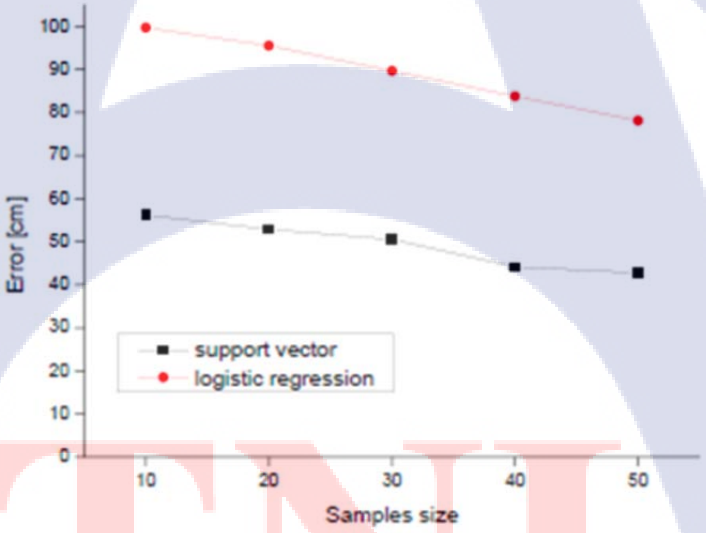
$$\mu_t = \bar{u}_t + K_t (z_t - \overline{\mu}_t) \quad (2.12)$$

$$\Sigma_t = \overline{\sum}_t - (K_t \overline{\sum}_t) \quad (2.13)$$

สมการที่ได้จากการแก้ไขเพื่อใช้ในการคำนวณตำแหน่งวัตถุที่หยุดนิ่งจะถูกนำไปใช้ในการทำหน้าที่ในการกรองค่าความคลาดเคลื่อนจากข้อมูลที่มีความแปรปรวนของการคำนวณที่ตำแหน่งส่งผลโดยตรงกับผลลัพธ์ [11]

2.2 งานวิจัยที่เกี่ยวข้อง

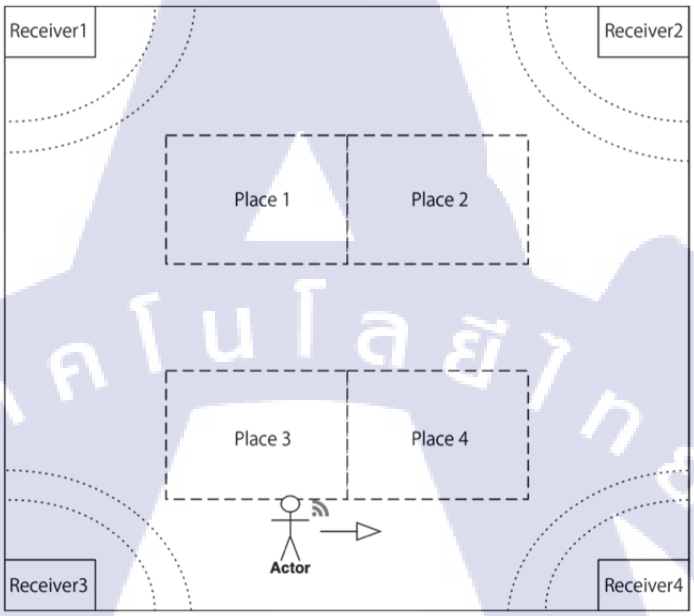
2.2.1 P. Sthapit et al. [12] งานวิจัยนี้ได้ทำการวิจัยเกี่ยวกับการหาตำแหน่งภายในอาคาร โดยใช้การเรียนรู้เครื่องจักร (Machine Learning : ML) ในการทำนายตำแหน่งของจุดที่เครื่องรับสัญญาณอยู่ ซึ่งในที่นี้คือโทรศัพท์มือถือ โดยวิธีการวิจัยถูกแบ่งเป็น 2 ช่วง คือ Offline Phase และ Online Phase โดยที่ Offline Phase คือช่วงของการเก็บข้อมูลเพื่อใช้ในการฝึกสอน โดยที่เก็บข้อมูล ณ จุดอ้างอิงต่างๆ ซึ่งตัวกระจายสัญญาณนั้นใช้ Beacon 14 ตัวในการกระจายสัญญาณ เมื่อได้ค่าความแรงสัญญาณ (Received Signal Strength : RSS) แล้วจึงนำไปฝึกสอนเพื่อให้ได้โมเดลที่สามารถทำนายผลออกมาเป็นค่าความน่าจะเป็นของจุดที่ตัวรับสัญญาณนั้นอยู่ โดยโมเดลที่นำมาใช้ในการฝึกสอนนั้นได้แก่ Support Vector Machine (SVM) และ Logistic Regression ส่วนช่วงของ Online Phase นั้นจะเป็นขั้นตอนการทดสอบ โดยการคำนวณค่าความคลาดเคลื่อนนั้น ใช้ Euclidean Distance โดยผลลัพธ์ของการทดลองพบว่า Support Vector Machine ให้ผลลัพธ์ที่ดีกว่า โดยค่าเฉลี่ยของความคลาดเคลื่อนอยู่ที่ 50 เซนติเมตร อีกทั้งการเพิ่มจำนวนข้อมูลที่ใช้ในการฝึกสอนก็ยิ่งทำให้ค่าความคลาดเคลื่อนลดลงอีกด้วย ซึ่งเป็นดังรูปที่ 2.5



รูปที่ 2.5 ค่าความคลาดเคลื่อนของระยะทางที่ทำนายกับระยะทางจริง ณ จำนวนข้อมูลที่ใช้ในการฝึกสอน

2.2.2 A. Sashida et al.[13] งานวิจัยนี้ได้ทำการวิจัยเกี่ยวกับระบบระบุตำแหน่งของโทรศัพท์มือถือด้วยเทคโนโลยีบลูทูธพลังงานต่ำ โดยการนำอัลกอริทึม Gradient Boosting Decision Tree (Xgboost) มาใช้ทำให้มีค่าความแม่นยำที่สูงมากยิ่งขึ้น ผู้วิจัยได้นำอุปกรณ์รับสัญญาณ BLEAD-B ซึ่งมีระยะรับสัญญาณไกลที่สุดที่ 20 เมตร โดยที่แต่ละตัวจะถูกติดตั้งที่มุมของห้องที่ใช้ในการทำวิจัยและ

ถูกตั้งสูงจากพื้นอยู่ที่ 50 เซนติเมตร ในการทำวิจัยผู้ทำวิจัยได้ใช้อุปกรณ์กระจายสัญญาณบลูทูธพลังงานต่ำใส่ไว้ที่กระเป๋าเสื้อและทำการเก็บข้อมูล 2 แบบ คือ แบบเคลื่อนที่ และหยุดนิ่งอยู่กับที่ตามรูปที่ 2.6



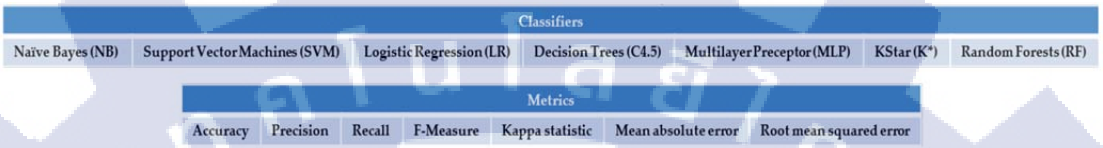
รูปที่ 2.6 พื้นที่ที่ใช้ในการทำวิจัย

จากงานวิจัยนี้พบว่าค่าความแรงของสัญญาณที่เก็บได้สำหรับการเก็บข้อมูลแบบเคลื่อนที่มีความคลาดเคลื่อนมากกว่าแบบหยุดนิ่ง อีกทั้งสิ่งกีดขวางภายในห้องก็ส่งผลต่อค่าความแรงของสัญญาณอีกด้วย ในส่วนของการนำอัลกอริทึมการเรียนรู้เครื่องจักรมาใช้นั้นผู้วิจัยได้ใช้วิธีการทดสอบ K-Cross-Validation โดยที่ k = 5 ซึ่งผลลัพธ์ในแต่ละรอบเป็นไปตามรูปที่ 2.7 และมีค่าความแม่นยำมากกว่า 90%

1st	0.93150685
2nd	0.93835616
3rd	0.91780822
4th	0.92465753
5th	0.89726027

รูปที่ 2.7 ผลลัพธ์ค่าความแม่นยำด้วยวิธีการทดสอบ K-Cross-Validation ทั้ง 5 รอบ

2.2.3 V. Stavrou et al. [14] งานวิจัยนี้ได้ทำการวิจัยเกี่ยวกับระบบระบุพิกัดภายในอาคารด้วยเทคโนโลยีบลูทูธพลังงานต่ำภายในร้านค้าปลีกที่มี 2 ชั้น ด้วยวิธีการ Fingerprinting ร่วมกับอัลกอริทึมการเรียนรู้เครื่องจักรและ Ensemble Filter มาใช้ในการระบุตำแหน่งของลูกค้า และทำการประเมินผลลัพธ์ด้วยการคำนวณ Accuracy, Precision, Recall, F-Measure, Kappa Statistic, Mean Absolute Error และ Root Mean Squared Error โดยที่อัลกอริทึมที่นำมาใช้ได้แก่ Naïve Bayes (NB), Support Vector Machines (SVM), Logistic Regression (LR), Decision Trees (C4.5), Multilayer Preceptor (MLP), KStar (K^*) และ Random Forests (RF) ตามรูปที่ 2.8



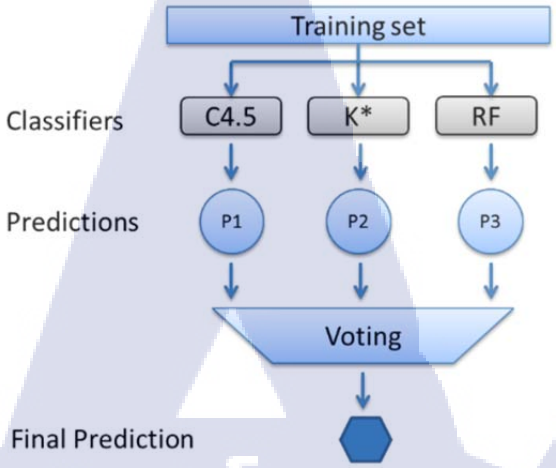
รูปที่ 2.8 อัลกอริทึมการเรียนรู้เครื่องจักรที่ใช้และวิธีการคำนวณผลลัพธ์เพื่อใช้ในการเปรียบเทียบ

งานวิจัยนี้ใช้วิธีการประเมินผล 10-Fold Cross-Validation ซึ่งทำการเปรียบเทียบผลลัพธ์ของแต่ละอัลกอริทึมตามตารางที่ 2.1

ตารางที่ 2.1 ผลลัพธ์โดยเฉลี่ยของการเปรียบเทียบการทำนายพิกัดแต่ละตำแหน่งของแต่ละอัลกอริทึม

Classification Algorithms (Classifiers)							
Assessment Metrics	NB	SVM	LR	C4.5	MLP	K*	RF
Accuracy	74.08% (1.73)	85.11% (1.48)	85.92% (1.70)	86.84% (1.36)	90.87% (2.01)	93.51% (1.33)	95.95% (1.06)
F-Measure	0.744 (0.10)	0.850 (0.08)	0.859 (0.09)	0.868 (0.06)	0.908 (0.13)	0.935 (0.05)	0.959 (0.03)
Kappa Statistic	0.7341 (0.02)	0.8477 (0.01)	0.8564 (0.02)	0.8651 (0.01)	0.9062 (0.01)	0.9336 (0.01)	0.9586 (0.01)
Mean Absolute Error	0.0101 (0.03)	0.0371 (0.02)	0.0056 (0.02)	0.0056 (0.01)	0.0043 (0.02)	0.0026 (0.01)	0.0086 (0.01)
Root Mean Squared Error	0.08891 (0.03)	0.1351 (0.02)	0.0592 (0.02)	0.0662 (0.01)	0.0543 (0.02)	0.0439 (0.01)	0.0478 (0.01)

จากผลลัพธ์ที่ได้ผู้วิจัยได้นำอัลกอริทึมที่ดีที่สุด 3 อัลกอริทึมมาทำการโหวตเพื่อทำนายพิกัดก่อนนำไปประเมินผลด้วยวิธีการคำนวณต่างๆ ตามรูปที่ 2.9 ซึ่ง Ensemble Filter ที่ใช้คือ Meta-Classifier ทำให้ผลลัพธ์สุดท้ายที่ได้ตามตารางที่ 2.2 ซึ่งทำการเปรียบเทียบแล้วพบว่าอัลกอริทึม Random Forest มีผลลัพธ์ที่ดีกว่า Ensemble Filter



รูปที่ 2.9 ขั้นตอนการเลือกอัลกอริทึมมาใช้ร่วมกับ Ensemble Filter

ตารางที่ 2.2 เปรียบเทียบผลลัพธ์ของ 3 อัลกอริทึมที่ดีที่สุดร่วมกับ Ensemble Filter

	Classifiers			
Assessment Metrics	**C4.5**	**K***	**RF**	**Ensemble**
Accuracy	86.84% (1.36)	93.51% (1.33)	95.95% (1.06)	95.78% (1.00)
F-Measure	0.868% (0.06)	0.935% (0.05)	0.959% (0.03)	0.957% (0.02)
Kappa Statistic	0.8651% (0.01)	0.9336% (0.01)	0.9586% (0.01)	0.9569% (0.01)
Mean Absolute Error	0.0056% (0.01)	0.0026% (0.01)	0.0086% (0.01)	0.0051% (0.01)
Root Mean Squared Error	0.0662% (0.01)	0.0439% (0.01)	0.0478% (0.01)	0.0392% (0.01)

บทที่ 3
ระเบียบวิธีการวิจัย

ในบทนี้กล่าวถึงระเบียบวิธีการวิจัยเกี่ยวกับระบบระบุตำแหน่งภายในอาคาร โดยใช้วิธีการเรียนรู้เครื่องจักรร่วมกับอัลกอริทึม Kalman Filter ซึ่งงานวิจัยนี้นำอุปกรณ์ที่ใช้ในการรับ-ส่งสัญญาณบลูทูธพลังงานต่ำ (Bluetooth Low Energy: BLE) เข้ามาใช้ในการเก็บข้อมูลด้วย Raspberry Pi 4 และตัวกระจายสัญญาณ Ruuvi เพื่อใช้ในการฝึกสอนและทดสอบโมเดลที่มีความแม่นยำที่ยอมรับได้ โดยเนื้อหาบทนี้จะกล่าวถึงวิธีการและขั้นตอนการดำเนินงานวิจัยดังต่อไปนี้

3.1 อุปกรณ์ที่ใช้ในการทดลอง

3.2 ขั้นตอนการดำเนินงาน

3.1 อุปกรณ์ที่ใช้ในการทดลอง

3.1.1 ตัวกระจายสัญญาณ Ruuvi

ในงานวิจัยนี้ได้ใช้ตัวกระจายสัญญาณ Ruuvi 5 ตัว ในการทดลอง ซึ่งในงานวิจัยนี้ได้ทำการติดตั้งอุปกรณ์นี้ที่มุมและตรงกลางของห้องที่ใช้ในการทำวิจัย โดยมีระยะความห่างจากพื้นห้องไม่เท่ากัน เนื่องจากจุดที่สามารถวางอุปกรณ์ได้นั้นมีระยะความสูงที่ต่างกัน

รูปที่ 3.1 ตัวกระจายสัญญาณ Ruuvi

3.1.2 อุปกรณ์ฝังตัว Raspberry Pi 4

ในงานวิจัยนี้ได้ใช้อุปกรณ์ฝังตัว Raspberry Pi 4 1 ตัว ซึ่งในขณะที่ทำการทดลองอุปกรณ์ฝังตัว Raspberry Pi 4 จะถูกบรรจุด้วยชุดคำสั่งที่ใช้ในการเก็บข้อมูลค่าความแรงของสัญญาณจากตัวกระจายสัญญาณทั้ง 5 จุดที่ถูกติดตั้งไว้ และ ใช้ในการทดสอบโมเดลที่ถูกฝึกสอนด้วยข้อมูลที่ถูกเก็บจากขั้นตอนการเก็บรวบรวมข้อมูลซึ่งบรรจุอยู่ในชุดคำสั่งที่ใช้ในขั้นตอนการทดสอบ โดยที่จะทำการเก็บข้อมูลในตำแหน่งต่างๆ จากอุปกรณ์กระจายสัญญาณทั้ง 5 แล้วทำการรวบรวมตามจำนวนที่ระบุไว้ก่อนจะนำไปประมวลผลเพื่อให้ได้ผลลัพธ์ที่สามารถนำไปเปรียบเทียบและสรุปผลการทดลองได้

รูปที่ 3.2 ภาพอุปกรณ์ฝังตัว Raspberry Pi 4

3.2 ขั้นตอนการดำเนินงาน

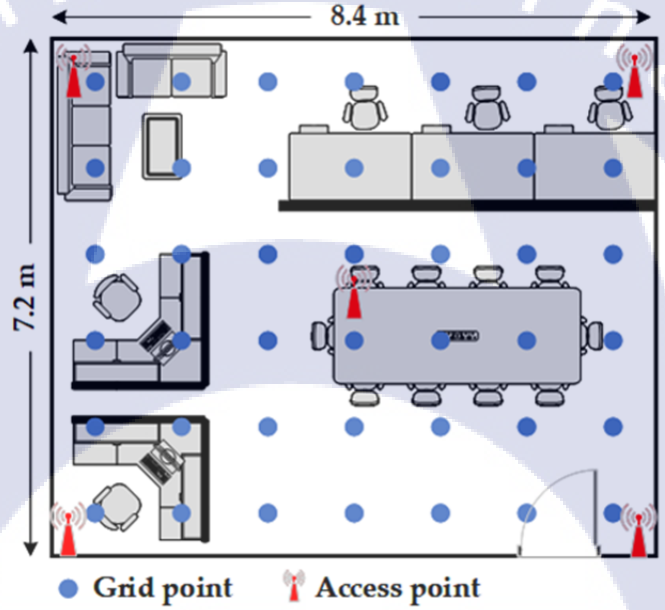
3.2.1 แผนภาพ Flowchart ขั้นตอนการดำเนินงาน

รูปที่ 3.3 Flowchart ขั้นตอนการดำเนินงาน

3.2.2 ขั้นตอนการออกแบบสถานที่ทำการวิจัย

ในส่วนของสถานที่ทำการวิจัยนั้นได้ทำการเลือกห้องที่มีขนาดความกว้าง 7.2 เมตร x ความยาว 8.4 เมตร โดยทำการแบ่งพื้นที่ออกเป็น 42 จุดตามรูปภาพที่ 3.4 ซึ่งแต่ละจุดจะถูกใช้อ้างอิงเป็นตำแหน่งของอุปกรณ์ในการระบุตำแหน่งว่าอุปกรณ์นั้นอยู่ที่ตำแหน่งใดของห้อง อีกทั้งอุปกรณ์กระจายสัญญาณจะถูกติดตั้งไว้ที่มุมทั้ง 4 ของห้อง และตรงกลางของห้องอีก 1 ตัว

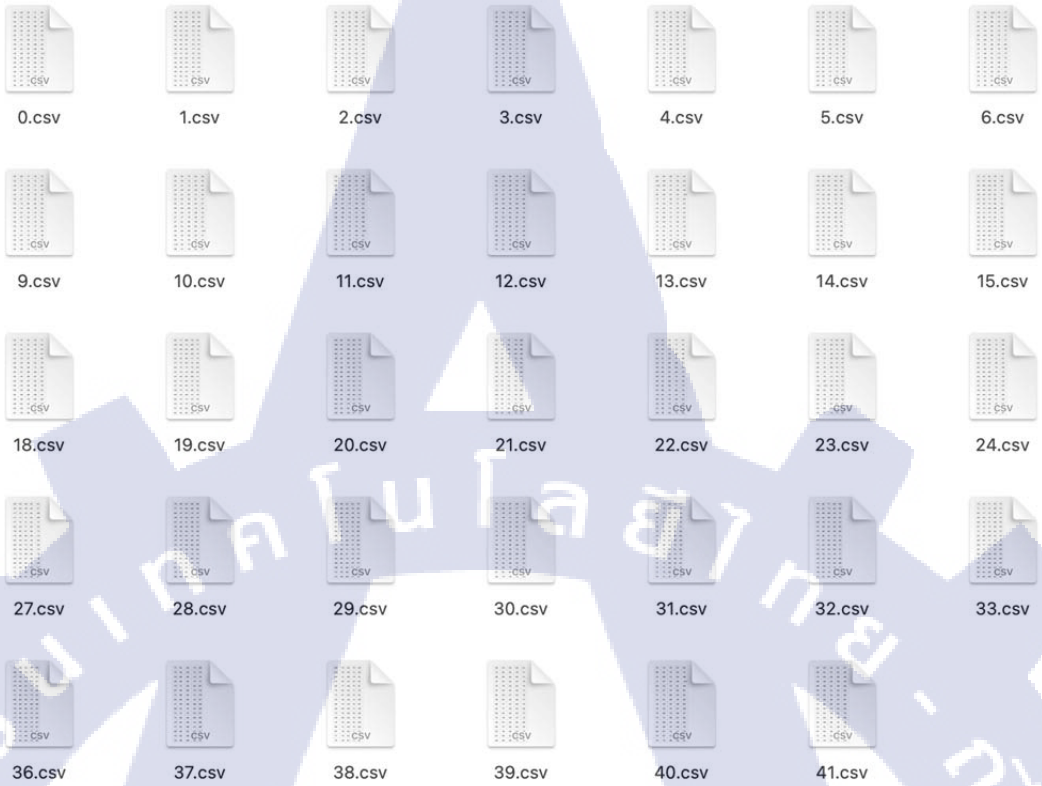
ภายในห้องที่ใช้ทำการทดลองจะประกอบไปด้วยเครื่องเรือนรวมถึงฉากที่ใช้ในการแบ่งส่วนของห้องซึ่งส่งผลต่อคุณภาพของสัญญาณ ดังนั้นจึงนำอุปกรณ์กระจายสัญญาณวางไว้สูง แต่เนื่องจากความสูงที่สามารถวางอุปกรณ์ได้นั้นมีระยะที่ไม่เท่ากันในแต่ละจุดทำให้ระยะความสูงของอุปกรณ์แต่ละตัวไม่เท่ากัน



รูปที่ 3.4 โครงสร้างของพื้นที่ที่ใช้ในการทดลอง

3.2.3 ขั้นตอนการเก็บข้อมูล

หลังจากการออกแบบสถานที่ทำการทดลอง ตำแหน่งทั้ง 42 จุดจะถูกใช้ในการเก็บรวบรวมข้อมูลเพื่อใช้ในการฝึกสอนและทดสอบทำให้จำนวนข้อมูลที่เก็บได้จะมีทั้งหมด 42 ชุดตามจุดที่ใช้ในการอ้างอิงและมีข้อมูลสัญญาณทั้งหมด 5 ตัวซึ่งอ้างอิงจากจำนวนของอุปกรณ์กระจายสัญญาณ



รูปที่ 3.5 ชุดข้อมูลได้จากการเก็บข้อมูลค่าความแรงของสัญญาณในรูปแบบ csv

ซึ่งการเก็บข้อมูลนั้นจะใช้อุปกรณ์ฝังตัว Raspberry Pi 4 ที่ถูกบรรจุชุดคำสั่งภาษา Python ที่ใช้ Library bluepy ในการเขียนคำสั่งให้อุปกรณ์สามารถเก็บข้อมูลค่าความแรงของสัญญาณจากอุปกรณ์กระจายสัญญาณทั้ง 5 ตัว ด้วยเทคโนโลยีบลูทูธพลังงานต่ำ โดยที่ชุดคำสั่งที่ใช้เป็นดังรูปที่ 3.6

```python
from bluepy.btle import Scanner, DefaultDelegate
import pandas as pd
import sys
import os
import getopt
import time

class ScanDelegate(DefaultDelegate):
    def __init__(self):
        DefaultDelegate.__init__(self)

    def handleDiscovery(self, dev, isNewDev, isNewData):
        if isNewDev:
            print("Discovered device", dev.addr)
        elif isNewData:
            print("Received new data from", dev.addr)

scanner = Scanner()

def loop(number, scanTime, labelName):
    dir_path = 'rawdata'
    if not os.path.isdir(dir_path):
        os.makedirs(dir_path)
    for i in range(number):
        print("round {0}".format(i+1))
        devices = scanner.scan(float(scanTime))
        rssi1 = -100
        rssi2 = -100
        rssi3 = -100
        rssi4 = -100
        rssi5 = -100
        for dev in devices:
            if dev.addr == "FC:F5:CB:EB:BA:AD".lower(): # BAAD
                rssi1 = dev.rssi
            elif dev.addr == "CC:E0:9A:F7:25:59".lower(): # 2559
                rssi2 = dev.rssi
            elif dev.addr == "DE:56:DD:68:B7:41".lower(): # B741
                rssi3 = dev.rssi
            elif dev.addr == "F9:86:4B:8B:C4:A7".lower(): # C4A7
                rssi4 = dev.rssi
            elif dev.addr == "FF:C6:1D:F7:30:E9".lower(): # 30E9
                rssi5 = dev.rssi

        df = pd.DataFrame({ "AC1": [rssi1],
                            "AC2": [rssi2],
                            "AC3": [rssi3],
                            "AC4": [rssi4],
                            "AC5": [rssi5], })
        df.to_csv(os.path.join(dir_path, labelName)+".csv", mode='a', index=False, header=False)

def main(argv):
    arg_help = "{0} -n <number> -s <scanTime> -l <labelName>".format(argv[0])
    try:
        opts, _ = getopt.getopt(argv[1:], "hn:s:l:", ["help", "number=", "scanTime=", "labelName="])
    except:
        print(arg_help)
        sys.exit(2)

    number = 3
    scanTime = 2
    labelName = ""
    for opt, arg in opts:
        if opt in ("-h", "--help"):
            print(arg_help)
            sys.exit(2)
        elif opt in ("-n", "--number"):
            number = int(arg)
        elif opt in ("-s", "--scanTime"):
            scanTime = int(arg)
        elif opt in ("-l", "--labelName"):
            labelName = arg
    if labelName == "":
        print(arg_help)
        sys.exit(2)
    loop(number, scanTime, labelName)

if __name__ == "__main__":
    main(sys.argv)
```

รูปที่ 3.6 ชุดคำสั่งที่ใช้ในการเก็บข้อมูลที่ถูกบรรจุในอุปกรณ์ฝังตัว Raspberry Pi 4

3.2.4 ขั้นตอนการเลือกใช้อัลกอริทึม

ในส่วนนี้จะทำการเลือกใช้โมเดลจาก Library Scikit-Learn เพื่อนำมาใช้ในการฝึกสอนโมเดลที่เขียนคำสั่งด้วยภาษาด้วยภาษา Python ซึ่งโมเดลที่ถูกเลือกนำมาใช้ได้แก่ K-nearest Neighbors Classification, Support Vector Classification, Random Forest และ Naïve Bayes

แต่เนื่องจากข้อมูลค่าความแรงสัญญาณที่ได้นั้นมีความคลาดเคลื่อนสูง จึงมีการนำ Kalman Filter มาใช้งานเพื่อช่วยในการจัดระเบียบข้อมูลให้อยู่ในช่วงเดียวกัน ซึ่งในงานวิจัยนี้ได้ทำการเขียนคำสั่งฟังก์ชั่นของอัลกอริทึมนี้ตามรูปที่ 3.7

```python
class KalmanFilter:
  def __init__(self, R = 1, Q = 1, A = 1, B = 0, C = 1):
    self.R = R #noise power desirable (process noise)
    self.Q = Q #noise power estimated (measurement noise)

    self.A = A;
    self.C = C;
    self.B = B;
    self.cov = None;
    self.x = None; #estimated signal without noise

  #z = Measurement, u = Control
  def filter(self, z, u = 0):
    if self.x == None:
      self.x = (1 / self.C) * z
      self.cov = (1 / self.C) * self.Q * (1 / self.C)
    else:
      predX = self.predict(u)
      predCov = self.uncertainty()

      K = predCov * self.C * (1 / ((self.C * predCov * self.C) + self.Q))

      self.x = predX + K * (z - (self.C * predX))
      self.cov = predCov - (K * self.C * predCov)
    return self.x

  def predict(self, u = 0):
    return (self.A * self.x) + (self.B * u) #A == F

  def uncertainty(self):
    return ((self.A * self.cov) * self.A) + self.R #A == F, cov == P

  def lastMeasurement(self):
    return self.x

  def setMeasurementNoise(self, noise):
    self.Q = noise

  def setProcessNoise(self, noise):
    self.R = noise
```

รูปที่ 3.7 อัลกอริทึม Kalman Filter ที่ถูกเขียนฟังก์ชั่นด้วยภาษา Python

3.2.5 ขั้นตอนการฝึกสอนโมเดล

ในส่วนแรกจะนำข้อมูลมาผ่านวิธีการ Normalization โดยใช้อัลกอริทึม Kalman Filter เพื่อที่จะปรับค่าความแรงของสัญญาณที่ถูกเก็บรวบรวมมานั้นมีความใกล้เคียงกันของทุกๆ ตัวอย่าง โดยรูปที่ 3.8 แสดงถึงฟังก์ชั่นที่ใช้ในการกรองค่าความแรงของสัญญาณจากอุปกรณ์กระจายสัญญาณทั้ง 5 ตัว ซึ่ง Kalman Filter ที่ใช้จะใช้ค่า Process Noise เท่ากับ 0.01 และ Measurement Noise เท่ากับ 3 เนื่องจากค่าความคลาดเคลื่อนของเครื่องที่ใช้ในการเก็บข้อมูลค่อนข้างสูงจึงมีการกำหนดค่าให้มีค่ามาก และในความเป็นจริงสภาพของอุปกรณ์ที่ใช้ในการเก็บข้อมูลนั้นจะถูกตั้งไว้อยู่กับที่ทำให้ค่าของ Process Noise ที่กำหนดมีค่าที่น้อยและไม่มีการกำหนดค่า State Transition ซึ่งแสดงอยู่ในรูปที่ 3.8

```
def kalman5Attr(df):
    tmp_df = df
    for i in range(5):
        kf = KalmanFilter(R=0.01, Q=3)
        tmp_list = []
        for rssi in tmp_df[i]:
            tmp_list.append(kf.filter(rssi))
        tmp_df[i] = tmp_list
    return tmp_df
```

รูปที่ 3.8 ฟังก์ชั่นที่ใช้ในการกรองข้อมูลค่าความแรงของสัญญาณจากอุปกรณ์กระจายสัญญาณทั้ง 5 ตัว ด้วย Kalman Filter

ก่อนที่จะเข้าสู่กระบวนการ Normalization จะต้องทำการดึงข้อมูลจากไฟล์ csv เพื่อที่จะเรียกใช้ข้อมูลที่เก็บรวบรวมมา อีกทั้งงานวิจัยนี้จำเป็นที่จะต้องทำการเปรียบเทียบผลลัพธ์ระหว่างโมเดลที่ถูกฝึกสอนด้วยชุดข้อมูลที่ผ่านและไม่ผ่าน Kalman Filter ดังนั้นจึงมีการจัดเตรียมชุดข้อมูลไว้ 2 แบบ คือ แบบไม่ Normalize และแบบ Normalize โดยคำสั่งที่ใช้ในการดึงข้อมูลและทำการ Normalize ข้อมูลจะเป็นไปตามรูปที่ 3.9 ซึ่งข้อมูลที่ดึงมาจะประกอบไปด้วยตำแหน่งที่เก็บข้อมูลทั้งหมด 42 ตำแหน่งและแต่ละตำแหน่งจะมีข้อมูลของอุปกรณ์กระจายสัญญาณทั้ง 5 ตัว โดยจำนวนตัวอย่างที่ใช้ในการฝึกสอนประมาณไม่น้อยกว่า 750 ตัวอย่างต่อตำแหน่ง

```
ROOT_PATH = './rawdata'


def genInputAndLabel(labelNum, withNormalize = False):
  input = pd.read_csv('{}/{}.csv'.format(ROOT_PATH, labelNum), header=None)
  trainData = input
  if withNormalize == True:
    trainData = kalman5Attr(trainData)
  trainLabel = [int(labelNum)] * trainData.shape[0]
  return trainData.sample(frac=1).reset_index(drop=True), trainLabel


NUM_CLASS = 42


unnormalizeInput = pd.DataFrame()
unnormalizeLabel = []
input2 = pd.DataFrame()
label2 = []

for i in range(NUM_CLASS):
  data, label = genInputAndLabel(i)
  unnormalizeInput = pd.concat([unnormalizeInput, data])
  unnormalizeLabel = np.concatenate([unnormalizeLabel, label])

  dataKalman, labelKalman = genInputAndLabel(i, True)
  input2 = pd.concat([input2, dataKalman])
  label2 = np.concatenate([label2, labelKalman])
```

รูปที่ 3.9 ฟังก์ชั่นที่ใช้ในการดึงข้อมูลจากไฟล์ csv เพื่อเปลี่ยนให้ข้อมูลอยู่ในรูปแบบที่สามารถใช้ในการฝึกสอนข้อมูลได้

เมื่อชุดข้อมูลที่ผ่านและไม่ผ่านการ Normalization แล้วจะถูกนำไปใช้ในการฝึกสอนโมเดลโดยที่ชุดคำสั่งที่ใช้ในการฝึกสอนจะเป็นไปตามรูปที่ 3.10-3.13 ซึ่งจะทำการฝึกสอน 2 แบบ และถูกบันทึกเป็นไฟล์ด้วย Library Joblib ที่ใช้สำหรับการบันทึกโมเดลเพื่อที่จะนำไปใช้ในส่วนของการทดสอบและประเมินวัดผลลัพธ์ของการวิจัย ดังนั้นโมเดลทั้งหมดที่จะได้จากขั้นตอนนี้จะมีทั้งหมด 8 โมเดลที่ใช้ในการเปรียบเทียบและสรุปผลการทดลอง

```
KNN K=8


def getKNNModelandPrediction(train_x, train_y):
  knn_clf=KNeighborsClassifier(n_neighbors = 8)
  st = time.time()
  knn_clf.fit(train_x, train_y)
  et = time.time()
  print('Training time: ', et - st, ' seconds')

  return knn_clf


KNNModel = getKNNModelandPrediction(unnormalizeInput, unnormalizeLabel)
joblib.dump(KNNModel, "../model/KNN.joblib")

kalmanKNNModel = getKNNModelandPrediction(input2, label2)
joblib.dump(KNNModel, "../model/KNN_KALMAN.joblib")
```

รูปที่ 3.10 ชุดคำสั่งที่ใช้ในการฝึกสอนและบันทึกโมเดลจากอัลกอริทึม K-nearest Neighbors ที่ค่า k เท่ากับ 8

```
SVC


def getSVCModelandPrediction(train_x, train_y):
  svc_clf = make_pipeline(StandardScaler(), SVC(gamma='auto'))
  st = time.time()
  svc_clf.fit(train_x, train_y)
  et = time.time()
  print('Training time: ', et - st, ' seconds')

  return svc_clf


SVCModel = getSVCModelandPrediction(unnormalizeInput, unnormalizeLabel)
joblib.dump(SVCModel, "../model/SVC.joblib")

kalmanSVCModel = getSVCModelandPrediction(input2, label2)
joblib.dump(kalmanSVCModel, "../model/SVC_KALMAN.joblib")
```

รูปที่ 3.11 ชุดคำสั่งที่ใช้ในการฝึกสอนและบันทึกโมเดลจากอัลกอริทึม Support Vector Classification

```
Random Forest


def getRFModelandPrediction(train_x, train_y):
  rf_clf=RandomForestClassifier()
  st = time.time()
  rf_clf.fit(train_x, train_y)
  et = time.time()
  print('Training time: ', et - st, ' seconds')

  return rf_clf


RFModel = getRFModelandPrediction(unnormalizeInput, unnormalizeLabel)
joblib.dump(RFModel, "../model/RF.joblib")

kalmanRFModel = getRFModelandPrediction(input2, label2)
joblib.dump(kalmanRFModel, "../model/RF_KALMAN.joblib")
```

รูปที่ 3.12 ชุดคำสั่งที่ใช้ในการฝึกสอนและบันทึกโมเดลจากอัลกอริทึม Random Forest

```
Naïve Bayes


def getNBModelandPrediction(train_x, train_y):
  gnb = GaussianNB()
  st = time.time()
  gnb.fit(train_x, train_y)
  et = time.time()
  print('Training time: ', et - st, ' seconds')

  return gnb


NBModel = getNBModelandPrediction(unnormalizeInput, unnormalizeLabel)
joblib.dump(NBModel, "../model/NB.joblib")

kalmanNBModel = getNBModelandPrediction(input2, label2)
joblib.dump(kalmanNBModel, "../model/NB_KALMAN.joblib")
```

รูปที่ 3.13 ชุดคำสั่งที่ใช้ในการฝึกสอนและบันทึกโมเดลจากอัลกอริทึม Naïve Bayes

3.2.6 ขั้นตอนการทดสอบระบบ

ในส่วนของการทดสอบนั้นจะทำการบรรจุคำสั่งที่ใช้ในการเก็บข้อมูลค่าความแรงสัญญาณจากอุปกรณ์กระจายสัญญาณเช่นเดียวกับขั้นตอนในการเก็บข้อมูลในส่วนแรก แต่จะทำการเก็บข้อมูลและรวบรวมชุดตามจำนวนที่กำหนดค่าไว้ โดยงานวิจัยนี้ได้กำหนดค่าในการเก็บข้อมูลชุดทดสอบอยู่ที่จุดละประมาณไม่น้อยกว่า 250 ตัวอย่าง เมื่อทำการเก็บข้อมูลตัวอย่างเรียบร้อยแล้วก็จะนำข้อมูลที่ได้แบ่งออกเป็น 2 ชุด ซึ่งชุดแรกคือแบบที่ไม่ผ่าน Kalman Filter และแบบที่สองคือแบบที่ผ่าน Kalman Filter โดยที่แบบแรกจะถูกใช้คู่กับโมเดลแบบที่ฝึกสอนด้วยชุดข้อมูลที่ไม่ผ่าน Kalman Filter ในส่วนของแบบที่สองหลังจากผ่าน Kalman Filter แล้วก็จะนำข้อมูลที่ได้ใช้คู่กับโมเดลที่ฝึกสอนด้วยข้อมูลที่ผ่าน Kalman Filter เพื่อใช้ในการทำนายผลและบันทึกผลลัพธ์ลงในไฟล์ csv เพื่อใช้ในการสรุปผลลัพธ์และทำการเปรียบเทียบด้วยวิธีการคำนวณแบบต่างๆ คำสั่งที่ใช้ในส่วนนี้เป็นไปตามรูปที่ 3.14-3.16

```
knnModel = joblib.load("../model/KNN.joblib")
knnKalmanModel = joblib.load("../model/KNN_KALMAN.joblib")
def KNN(x, kalman = False):
  if kalman == True:
    model = knnModel
  else:
    model = knnKalmanModel
  pred = model.predict(x)
  return pred

svcModel = joblib.load("../model/SVC.joblib")
svcKalmanModel = joblib.load("../model/SVC_KALMAN.joblib")
def SVC(x, kalman = False):
  if kalman == True:
    model = svcModel
  else:
    model = svcKalmanModel
  pred = model.predict(x)
  return pred

rfModel = joblib.load("../model/RF.joblib")
rfKalmanModel = joblib.load("../model/RF_KALMAN.joblib")
def RF(x, kalman = False):
  if kalman == True:
    model = rfModel
  else:
    model = rfKalmanModel
  pred = model.predict(x)
  return pred

nbModel = joblib.load("../model/NB.joblib")
nbKalmanModel = joblib.load("../model/NB_KALMAN.joblib")
def NB(x, kalman = False):
  if kalman == True:
    model = nbModel
  else:
    model = nbKalmanModel
  pred = model.predict(x)
  return pred
```

รูปที่ 3.14 ฟังก์ชั่นที่ใช้ในการดึงโมเดลที่ฝึกสอนด้วยด้วยอัลกอริทึมต่างๆ จากไฟล์และทำนายตำแหน่งของอุปกรณ์

```python
def loop(number, scanTime, labelName):
    dir_path = 'result'
    if not os.path.isdir(dir_path):
        os.makedirs(dir_path)

    rssi1List = []
    rssi2List = []
    rssi3List = []
    rssi4List = []
    rssi5List = []
    for i in range(number):
        print("round {0}".format(i+1))
        devices = scanner.scan(float(scanTime))
        rssi1 = -100
        rssi2 = -100
        rssi3 = -100
        rssi4 = -100
        rssi5 = -100
        for dev in devices:
            if dev.addr == "FC:F5:CB:EB:BA:AD".lower(): # BAAD
                rssi1 = dev.rssi
            elif dev.addr == "CC:E0:9A:F7:25:59".lower(): # 2559
                rssi2 = dev.rssi
            elif dev.addr == "DE:56:DD:68:B7:41".lower(): # B741
                rssi3 = dev.rssi
            elif dev.addr == "F9:86:4B:8B:C4:A7".lower(): # C4A7
                rssi4 = dev.rssi
            elif dev.addr == "FF:C6:1D:F7:30:E9".lower(): # 30E9
                rssi5 = dev.rssi
        rssi1List.append(rssi1)
        rssi2List.append(rssi2)
        rssi3List.append(rssi3)
        rssi4List.append(rssi4)
        rssi5List.append(rssi5)
    df = pd.DataFrame({ "AC1": rssi1List,
                        "AC2": rssi2List,
                        "AC3": rssi3List,
                        "AC4": rssi4List,
                        "AC5": rssi5List, })
    df = df.sample(frac=1).reset_index(drop=True)

    unnormalizeInput = df
    unnormalizeLabel = [int(labelName)] * number

    knn = KNN(unnormalizeInput)
    svc = SVC(unnormalizeInput)
    rf = RF(unnormalizeInput)
    nb = NB(unnormalizeInput)

    df = pd.DataFrame({ "KNN": knn,
                        "SVC": svc,
                        "RF": rf,
                        "NB": nb,
                        "REAL": unnormalizeLabel, })
    df.to_csv(os.path.join(dir_path, labelName)+"_without_kalman.csv", mode='a', index=False, header=True)

    input2 = kalmanSAttr(df)
    label2 = [int(labelName)] * number

    knn2 = KNN(input2, True)
    svc2 = SVC(input2, True)
    rf2 = RF(input2, True)
    nb2 = NB(input2, True)

    df = pd.DataFrame({ "KNN": knn2,
                        "SVC": svc2,
                        "RF": rf2,
                        "NB": nb2,
                        "REAL": label2, })
    df.to_csv(os.path.join(dir_path, labelName)+"_with_kalman.csv", mode='a', index=False, header=True)
```

รูปที่ 3.15 ชุดคำสั่งที่ใช้ในการเก็บข้อมูลชุดตัวอย่างและทำนายผลที่ถูกบันทึกลงในไฟล์ csv

```python
def main(argv):
    arg_help = "{0} -n <number> -s <scanTime> -l <labelName>".format(argv[0])
    try:
        opts, _ = getopt.getopt(argv[1:], "hn:s:l:", ["help", "number=", "scanTime=", "labelName="])
    except:
        print(arg_help)
        sys.exit(2)

    number = 3
    scanTime = 2
    labelName = ""
    for opt, arg in opts:
        if opt in ("-h", "--help"):
            print(arg_help)
            sys.exit(2)
        elif opt in ("-n", "--number"):
            number = int(arg)
        elif opt in ("-s", "--scanTime"):
            scanTime = int(arg)
        elif opt in ("-l", "--labelName"):
            labelName = arg
    if labelName == "":
        print(arg_help)
        sys.exit(2)
    loop(number, scanTime, labelName)

if __name__ == "__main__":
    main(sys.argv)
```

รูปที่ 3.16 ชุดคำสั่งที่ใช้ในการเรียกใช้งานฟังก์ชั่นที่ใช้ในการเก็บข้อมูลและทำนายผล

บทที่ 4
ผลการวิจัย

จากการศึกษาแนวคิด ทฤษฎี และงานวิจัยต่างๆ ที่เกี่ยวข้องเพื่อเป็นองค์ความรู้ในการศึกษาและวิจัยระบบระบุพิกัดภายในอาคาร โดยใช้วิธีการเรียนรู้เครื่องจักรร่วมกับอัลกอริทึม Kalman Filter ผู้วิจัยได้ทำการทดลองและวัดผลลัพธ์ของระบบด้วยขั้นตอนต่างๆ เพื่อที่จะสามารถพัฒนาระบบที่สามารถใช้งานได้จริง ซึ่งในส่วนนี้จะแสดงผลลัพธ์ของการทดลองโดยประกอบด้วยผลทำนายของโมเดลแต่ละโมเดลเปรียบเทียบกับพิกัดจริง

4.1 ภาพรวมของผลลัพธ์จากการทดลอง

ในส่วนภาพรวมของงานวิจัยนั้นถูกแบ่งออกเป็น 2 ขั้นตอน ได้แก่ ขั้นตอนการเก็บข้อมูลและฝึกสอนโมเดล และ ขั้นตอนการทดสอบ โดยที่พื้นที่ที่ใช้ในการทดลองและทดสอบใช้สถานที่เดียวกัน ในส่วนขั้นตอนการเก็บข้อมูลได้ทำการติดตั้งอุปกรณ์กระจายสัญญาณทั้ง 5 ตัวตามที่ได้ออกแบบไว้ ซึ่งใช้อุปกรณ์กระจายสัญญาณ Ruuvi ทั้งหมด และ ใช้อุปกรณ์ฝังตัว Raspberry Pi 4 ที่บรรจุชุดคำสั่งในการเก็บข้อมูลค่าความแรงของสัญญาณจากอุปกรณ์กระจายสัญญาณ ณ จุดต่างๆ ทั้งสิ้น 42 จุด หลังจากนั้นจึงนำข้อมูลที่ได้ไปทำการฝึกสอน ซึ่งการฝึกสอนโมเดลจะถูกแบ่งออกเป็น 2 แบบคือ ฝึกสอนโมเดลด้วยอัลกอริทึมการเรียนรู้เครื่องจักรเพียงอย่างเดียวและฝึกสอนด้วยอัลกอริทึมการเรียนรู้เครื่องจักรร่วมกับ Kalman Filter โดยอัลกอริทึมการเรียนรู้เครื่องจักรที่ใช้นั้นได้แก่ K-nearest neighbors Classification (K=8), Support Vector Classification, Random Forest (จำนวนต้นไม้=100) และ Naïve Bayes (รูปแบบอัลกอริทึมที่ใช้ในงานวิจัยทั้งหมดแสดงอยู่ในตารางที่ 4.1) หลังจากที่ได้โมเดลแล้วจึงนำโมเดลนั้นไปทดสอบและทำการเปรียบเทียบผลลัพธ์ของแต่ละโมเดลโดยการบรรจุชุดคำสั่งลงในอุปกรณ์ฝังตัว Raspberry Pi 4 เพื่อให้อุปกรณ์นำโมเดลไปใช้ในการทำนายผลและบันทึกผลลัพธ์ของการทดลองแล้วจึงนำผลลัพธ์นั้นไปใช้ในส่วนของการสรุปผลการทดลอง

ตารางที่ 4.1 รูปแบบอัลกอริทึมที่ใช้ในการทดลอง

Traditional	Support Vector Classification (SVC)
	Random Forest (RF) (Number of trees=100)
	Gaussian Naïve Bayes (NB)
	K-Nearest Neighbors (KNN) (K=8)

ตารางที่ 4.1 รูปแบบอัลกอริทึมที่ใช้ในการทดลอง (ต่อ)

Proposed	SVC + Kalman Filter
	RF + Kalman Filter
	NB + Kalman Filter
	KNN + Kalman Filter

4.2 ผลลัพธ์จากการฝึกสอนโมเดล

หลังจากที่ฝึกสอนโมเดลด้วยข้อมูลค่าความแรงสัญญาณที่ถูกเก็บรวบรวมแล้ว โดยที่ข้อมูลไม่ถูกประมวลผลผ่าน Kalman Filter ในส่วนของข้อมูลทดสอบที่ใช้นั้นจะไม่ถูกประมวลผลด้วย Kalman Filter เช่นกัน เนื่องจากต้องการให้ข้อมูลมีความสอดคล้องกันซึ่งผลลัพธ์ของโมเดลด้วยการคำนวณด้วยสูตรคำนวณได้แก่ Precision, Recall, F1-Score โดยที่จะแสดงให้เห็นในตารางที่ 4.2-4.5 สำหรับโมเดลที่ไม่ประมวลผลข้อมูลด้วย Kalman Filter อีกทั้งค่า Support ระบุถึงจำนวนตัวอย่างของตำแหน่งนั้นๆ ที่ใช้ในการทดสอบ รวมถึงแต่ละตารางจะแสดงค่าความแม่นยำของแต่ละโมเดลอีกด้วย

ตารางที่ 4.2 คะแนนของการฝึกสอนโมเดลด้วยอัลกอริทึม Support Vector Classification

Class	Precision	Recall	F1-Score	Support
0	0.82170543	0.848	0.83464567	250
1	0.83823529	0.912	0.87356322	250
2	0.83035714	0.744	0.78481013	250
3	0.56578947	0.688	0.62093863	250
4	0.24637681	0.204	0.22319475	250
5	0.23611111	0.136	0.17258883	250
6	0.76777251	0.648	0.70281996	250
7	0.58064516	0.648	0.61247637	250
8	0.71893491	0.972	0.82653061	250
9	0.56363636	0.496	0.52765957	250
10	0.4625323	0.716	0.56200942	250
11	0.19277108	0.064	0.0960961	250
12	0.5658363	0.636	0.59887006	250
13	0.27777778	0.22	0.24553571	500
14	0.41158537	0.54	0.46712803	250

ตารางที่ 4.2 คะแนนของการฝึกสอนโมเดลด้วยอัลกอริทึม Support Vector Classification (ต่อ)

Class	Precision	Recall	F1-Score	Support
15	0.51515152	0.952	0.66853933	250
16	0.52142857	0.292	0.37435897	250
17	0.42	0.616	0.49945946	375
18	0.3019943	0.212	0.24911868	500
19	0.42633929	0.764	0.54727794	250
20	0.48333333	0.232	0.31351351	250
21	0.63070539	0.608	0.6191446	250
22	0.51118211	0.64	0.56838366	250
23	0.06626506	0.044	0.05288462	250
24	0.35746606	0.316	0.33545648	250
25	0.66367713	0.78933333	0.72107186	375
26	0.39236111	0.38305085	0.38765009	295
27	0.41304348	0.65866667	0.50770812	375
28	0.38830898	0.496	0.43559719	375
29	0.40797546	0.35466667	0.37945792	375
30	0.62931035	0.584	0.60580913	375
31	0.27234043	0.17066667	0.20983607	375
32	0.43434343	0.344	0.38392857	375
33	0.78717201	0.72	0.75208914	375
34	0.36742424	0.25866667	0.30359937	375
35	0.51960784	0.848	0.6443769	375
36	0.68776371	0.652	0.66940452	250
37	0.48007246	0.62352941	0.54247697	425
38	0.23287671	0.04533333	0.07589286	375
39	0.46460177	0.28	0.34941764	375
40	0.32824428	0.22933333	0.2700157	375
41	0.59115044	0.89066667	0.7106383	375
Average	0.48510016	0.51133128	0.48395178	
Accuracy	0.49744177			

ตารางที่ 4.3 คะแนนของการฝึกสอนโมเดลด้วยอัลกอริทึม Random Forest

Class	Precision	Recall	F1-Score	Support
0	0.88842975	0.86	0.87398374	250
1	0.88518519	0.956	0.91923077	250
2	0.89903846	0.748	0.81659389	250
3	0.796875	0.816	0.80632411	250
4	0.49568966	0.46	0.47717842	250
5	0.71162791	0.612	0.65806452	250
6	0.85152838	0.78	0.81419624	250
7	0.77372263	0.848	0.80916031	250
8	0.81228669	0.952	0.87661142	250
9	0.68846154	0.716	0.70196078	250
10	0.72413793	0.84	0.77777778	250
11	0.39370079	0.4	0.3968254	250
12	0.74100719	0.824	0.78030303	250
13	0.60337553	0.572	0.58726899	500
14	0.66536965	0.684	0.67455621	250
15	0.69552239	0.932	0.7965812	250
16	0.64676617	0.52	0.57649667	250
17	0.66169154	0.70933333	0.68468469	375
18	0.47619048	0.36	0.41002278	500
19	0.54427083	0.836	0.65930599	250
20	0.65686275	0.268	0.38068182	250
21	0.64797508	0.832	0.72854641	250
22	0.67944251	0.78	0.72625698	250
23	0.128	0.064	0.08533333	250
24	0.4787234	0.36	0.4109589	250
25	0.84596577	0.92266667	0.88265306	375
26	0.6419214	0.49830509	0.5610687	295
27	0.55064457	0.79733333	0.65141612	375

ตารางที่ 4.3 คะแนนของการฝึกสอนโมเดลด้วยอัลกอริทึม Random Forest (ต่อ)

Class	Precision	Recall	F1-Score	Support
28	0.52783964	0.632	0.57524272	375
29	0.68435013	0.688	0.68617021	375
30	0.78	0.832	0.80516129	375
31	0.44444444	0.416	0.42975207	375
32	0.74200913	0.86666667	0.799508	375
33	0.88323353	0.78666667	0.83215797	375
34	0.38753388	0.38133333	0.3844086	375
35	0.78333333	0.87733333	0.82767296	375
36	0.86343612	0.784	0.82180294	250
37	0.6344464	0.84941177	0.72635815	425
38	0.36220472	0.12266667	0.18326693	375
39	0.48701299	0.4	0.43923865	375
40	0.52061856	0.53866667	0.52948886	375
41	0.83084577	0.89066667	0.85971686	375
Average	0.65513623	0.66697739	0.65295211	
Accuracy	0.65979381			

ตารางที่ 4.4 คะแนนของการฝึกสอนโมเดลด้วยอัลกอริทึม Gaussian Naïve Bayes

Class	Precision	Recall	F1-Score	Support
0	0.79051383	0.8	0.79522863	250
1	0.69846154	0.908	0.78956522	250
2	0.46428571	0.416	0.43881857	250
3	0.51502146	0.48	0.49689441	250
4	0.26229508	0.064	0.10289389	250
5	0.24	0.216	0.22736842	250
6	0.62704918	0.612	0.6194332	250
7	0.50574713	0.528	0.51663405	250
8	0.70175439	0.96	0.81081081	250

ตารางที่ 4.4 คะแนนของการฝึกสอนโมเดลด้วยอัลกอริทึม Gaussian Naïve Bayes (ต่อ)

Class	Precision	Recall	F1-Score	Support
9	0.47738694	0.38	0.42316258	250
10	0.27040816	0.212	0.23766816	250
11	0	0	0	250
12	0.42790698	0.368	0.39569893	250
13	0.2	0.056	0.0875	500
14	0.30063966	0.564	0.39221141	250
15	0.39726027	0.928	0.55635492	250
16	0.55789474	0.212	0.30724638	250
17	0.31746032	0.42666667	0.36405006	375
18	0.19461078	0.13	0.1558753	500
19	0.30906149	0.764	0.44009217	250
20	0.33067729	0.332	0.33133733	250
21	0.39529412	0.672	0.49777778	250
22	0.38864629	0.356	0.37160752	250
23	0.05747126	0.02	0.02967359	250
24	0.20858896	0.136	0.16464891	250
25	0.36510264	0.664	0.47114475	375
26	0.3699422	0.43389831	0.39937598	295
27	0.22469136	0.24266667	0.23333333	375
28	0.35835351	0.39466667	0.37563452	375
29	0.26053042	0.44533333	0.32874016	375
30	0.52529183	0.36	0.42721519	375
31	0.30392157	0.08266667	0.12997904	375
32	0.11695906	0.05333333	0.07326007	375
33	0.31650894	0.80266667	0.45399698	375
34	0.24404762	0.10933333	0.15101289	375
35	0.39206534	0.896	0.54545455	375
36	0.55384615	0.576	0.56470588	250

ตารางที่ 4.4 คะแนนของการฝึกสอนโมเดลด้วยอัลกอริทึม Gaussian Naïve Bayes (ต่อ)

Class	Precision	Recall	F1-Score	Support
37	0.49875312	0.47058824	0.4842615	425
38	0.20987654	0.04533333	0.0745614	375
39	0.23880597	0.08533333	0.12573674	375
40	0.13924051	0.02933333	0.04845815	375
41	0.51086957	0.62666667	0.56287425	375
Average	0.36350576	0.40139254	0.35719756	
Accuracy	0.38342879			

ตารางที่ 4.5 คะแนนของการฝึกสอนโมเดลด้วยอัลกอริทึม K-Nearest Neighbors

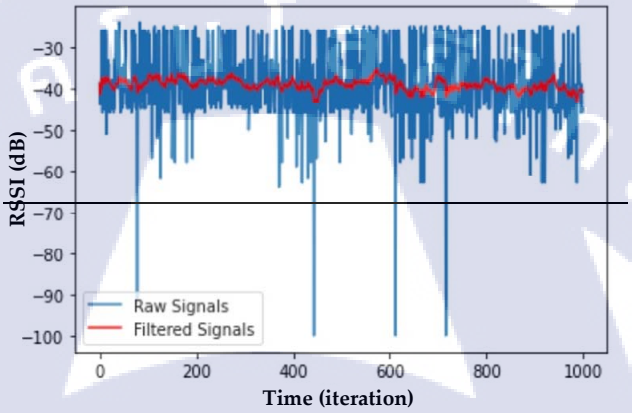
Class	Precision	Recall	F1-Score	Support
0	0.80933852	0.832	0.82051282	250
1	0.78911565	0.928	0.85294118	250
2	0.76209677	0.756	0.75903615	250
3	0.68041237	0.792	0.73197782	250
4	0.32270916	0.324	0.32335329	250
5	0.60619469	0.548	0.57563025	250
6	0.75720165	0.736	0.7464503	250
7	0.67368421	0.768	0.71775701	250
8	0.7516129	0.932	0.83214286	250
9	0.60207613	0.696	0.64564007	250
10	0.6552901	0.768	0.70718232	250
11	0.32432432	0.288	0.30508475	250
12	0.69111969	0.716	0.70333988	250
13	0.53441296	0.528	0.53118712	500
14	0.59770115	0.624	0.61056752	250
15	0.64343164	0.96	0.77046549	250
16	0.59281437	0.396	0.47482014	250
17	0.54772727	0.64266667	0.59141104	375

ตารางที่ 4.5 คะแนนของการฝึกสอนโมเดลด้วยอัลกอริทึม K-Nearest Neighbors (ต่อ)

Class	Precision	Recall	F1-Score	Support
18	0.35875706	0.254	0.29742389	500
19	0.4965358	0.86	0.6295754	250
20	0.6146789	0.268	0.37325905	250
21	0.6875	0.836	0.75451264	250
22	0.65917603	0.704	0.68085106	250
23	0.10606061	0.056	0.07329843	250
24	0.4127907	0.284	0.33649289	250
25	0.74097665	0.93066667	0.8250591	375
26	0.63181818	0.47118644	0.53980583	295
27	0.51865672	0.74133333	0.61031833	375
28	0.475	0.608	0.53333333	375
29	0.67857143	0.608	0.64135021	375
30	0.79508197	0.776	0.7854251	375
31	0.40728477	0.328	0.3633678	375
32	0.71229698	0.81866667	0.7617866	375
33	0.82939633	0.84266667	0.83597884	375
34	0.35384615	0.30666667	0.32857143	375
35	0.74130435	0.90933333	0.81676647	375
36	0.81027668	0.82	0.81510934	250
37	0.65730337	0.82588235	0.73201251	425
38	0.40145985	0.14666667	0.21484375	375
39	0.49751244	0.26666667	0.34722222	375
40	0.49544073	0.43466667	0.46306818	375
41	0.82451254	0.78933333	0.80653951	375
Average	0.601131	0.62191434	0.60155886	
Accuracy	0.61359297			

4.3 ผลลัพธ์จากการฝึกสอนโมเดลร่วมกับ Kalman Filter

ในส่วนของโมเดลที่ข้อมูลผ่านการประมวลผลด้วย Kalman Filter นั้นจะนำมาใช้ในส่วนของข้อมูลที่ใช้ฝึกสอนและข้อมูลที่ใช้ในการทดสอบ โดยเปรียบเทียบข้อมูลที่ประมวลผลด้วยอัลกอริทึม Kalman Filter กับข้อมูลดิบตามรูปที่ 4.1 จะพบว่าค่าความแรงของสัญญาณที่ประมวลผลด้วยอัลกอริทึม Kalman Filter (เส้นสีแดง) มีความเกาะกลุ่มมากกว่าข้อมูลแบบดิบ (เส้นสีน้ำเงิน) ซึ่งมีความกระจายตัวของข้อมูลมากกว่า ซึ่งผลลัพธ์ของการทดสอบโมเดลที่ประมวลผลข้อมูลด้วยอัลกอริทึม Kalman Filter จะแสดงให้เห็นในตารางที่ 4.6-4.9



รูปที่ 4.1 ผลลัพธ์ของการกรองค่าความแรงของสัญญาณด้วย Kalman Filter

ตารางที่ 4.6 คะแนนของการฝึกสอนโมเดลด้วยอัลกอริทึม Support Vector Classification ร่วมกับ Kalman Filter

Class	Precision	Recall	F1-Score	Support
0	1	0.996	0.99799599	250
1	0.98809524	0.996	0.99203187	250
2	0.99595142	0.984	0.98993964	250
3	1	1	1	250
4	0.98373984	0.968	0.97580645	250
5	0.8277512	0.692	0.75381264	250
6	1	1	1	250
7	1	0.996	0.99799599	250
8	0.99601594	1	0.99800399	250

ตารางที่ 4.6 คะแนนของการฝึกสอนโมเดลด้วยอัลกอริทึม Support Vector Classification ร่วมกับ Kalman Filter (ต่อ)

Class	Precision	Recall	F1-Score	Support
9	0.98814229	1	0.99403579	250
10	0.73287671	0.856	0.7896679	250
11	0.96442688	0.976	0.97017893	250
12	0.98814229	1	0.99403579	250
13	0.92565056	0.996	0.95953757	500
14	0.97129187	0.812	0.88453159	250
15	1	0.996	0.99799599	250
16	0.96595745	0.908	0.93608247	250
17	0.94117647	0.896	0.91803279	375
18	0.97911227	0.75	0.84937712	500
19	0.67213115	0.984	0.7987013	250
20	0.98969072	0.768	0.86486487	250
21	0.75465839	0.972	0.84965035	250
22	0.94921875	0.972	0.96047431	250
23	0.96174863	0.704	0.81293303	250
24	0.89873418	0.852	0.87474333	250
25	0.86310905	0.992	0.92307692	375
26	0.98961938	0.96949153	0.97945206	295
27	0.875	0.98933333	0.92866083	375
28	0.99447514	0.96	0.97693351	375
29	0.49800797	0.33333333	0.39936102	375
30	0.97883598	0.98666667	0.98273572	375
31	0.99159664	0.944	0.96721312	375
32	0.84965831	0.99466667	0.91646192	375
33	0.99463807	0.98933333	0.99197861	375
34	0.81958763	0.848	0.83355177	375
35	0.992	0.992	0.992	375

ตารางที่ 4.6 คะแนนของการฝึกสอนโมเดลด้วยอัลกอริทึม Support Vector Classification ร่วมกับ Kalman Filter (ต่อ)

Class	Precision	Recall	F1-Score	Support
36	0.984	0.984	0.984	250
37	0.69901316	1	0.82284608	425
38	0.50304878	0.44	0.46941679	375
39	0.96636086	0.84266667	0.9002849	375
40	0.84022039	0.81333333	0.82655827	375
41	0.99728997	0.98133333	0.98924731	375
Average	0.91216604	0.90795615	0.90581449	
Accuracy	0.9021764			

ตารางที่ 4.7 คะแนนของการฝึกสอนโมเดลด้วยอัลกอริทึม Random Forest ร่วมกับ Kalman Filter

Class	Precision	Recall	F1-Score	Support
0	1	0.988	0.99396378	250
1	1	0.996	0.99799599	250
2	0.99203187	0.996	0.99401198	250
3	0.99601594	1	0.99800399	250
4	0.99595142	0.984	0.98993964	250
5	0.85909091	0.756	0.80425532	250
6	1	1	1	250
7	0.99598394	0.992	0.99398798	250
8	0.98809524	0.996	0.99203187	250
9	0.99166667	0.952	0.97142857	250
10	0.78214286	0.876	0.82641509	250
11	0.85614035	0.976	0.91214953	250
12	0.97637795	0.992	0.98412698	250
13	0.90229885	0.942	0.92172211	500
14	0.86666667	0.728	0.79130435	250
15	1	0.976	0.98785425	250

ตารางที่ 4.7 คะแนนของการฝึกสอนโมเดลด้วยอัลกอริทึม Random Forest ร่วมกับ Kalman Filter (ต่อ)

Class	Precision	Recall	F1-Score	Support
16	0.94736842	0.864	0.90376569	250
17	0.91815857	0.95733333	0.93733682	375
18	0.97577093	0.886	0.92872117	500
19	0.8390411	0.98	0.90405904	250
20	0.97368421	0.74	0.84090909	250
21	0.70926518	0.888	0.78863233	250
22	0.94941634	0.976	0.96252466	250
23	0.94252874	0.656	0.77358491	250
24	0.86938776	0.852	0.86060606	250
25	0.91022444	0.97333333	0.94072165	375
26	0.99305556	0.96949153	0.98113208	295
27	0.85411765	0.968	0.9075	375
28	1	0.872	0.93162393	375
29	0.50387597	0.34666667	0.4107425	375
30	0.97619048	0.984	0.98007968	375
31	0.95833333	0.92	0.93877551	375
32	0.79700855	0.99466667	0.88493476	375
33	0.98670213	0.98933333	0.98801598	375
34	0.82939633	0.84266667	0.83597884	375
35	0.99166667	0.952	0.97142857	375
36	0.98	0.98	0.98	250
37	0.62684366	1	0.77062557	425
38	0.35409836	0.288	0.31764706	375
39	0.9380805	0.808	0.86819484	375
40	0.85674157	0.81333333	0.83447332	375
41	0.99726776	0.97333333	0.9851552	375
Average	0.90192112	0.89581329	0.89491335	
Accuracy	0.89095075			

ตารางที่ 4.8 คะแนนของการฝึกสอนโมเดลด้วยอัลกอริทึม Gaussian Naïve Bayes ร่วมกับ Kalman Filter

Class	Precision	Recall	F1-Score	Support
0	1	0.996	0.99799599	250
1	1	0.996	0.99799599	250
2	0.99196787	0.988	0.98997996	250
3	1	1	1	250
4	0.99593496	0.98	0.98790323	250
5	0.78076923	0.812	0.79607843	250
6	1	0.992	0.99598394	250
7	1	0.992	0.99598394	250
8	0.99601594	1	0.99800399	250
9	0.99193548	0.984	0.98795181	250
10	0.80082988	0.772	0.78615071	250
11	0.85964912	0.98	0.91588785	250
12	1	0.996	0.99799599	250
13	0.65671642	0.88	0.75213675	500
14	0.79824561	0.728	0.76150628	250
15	0.9469697	1	0.97276265	250
16	0.93951613	0.932	0.93574297	250
17	0.96782842	0.96266667	0.96524064	375
18	0.73044925	0.878	0.79745686	500
19	0.85714286	0.984	0.91620112	250
20	0.99082569	0.432	0.60167131	250
21	0.81270903	0.972	0.8852459	250
22	0.97222222	0.98	0.97609562	250
23	0.97826087	0.54	0.69587629	250
24	0.95798319	0.456	0.61788618	250
25	0.82017544	0.99733333	0.90012034	375
26	0.99530516	0.71864407	0.83464567	295

ตารางที่ 4.8 คะแนนของการฝึกสอนโมเดลด้วยอัลกอริทึม Gaussian Naïve Bayes ร่วมกับ Kalman Filter (ต่อ)

Class	Precision	Recall	F1-Score	Support
27	0.94642857	0.98933333	0.96740548	375
28	1	0.91733333	0.95688456	375
29	0.67181467	0.464	0.5488959	375
30	0.97568389	0.856	0.91193182	375
31	0.97484277	0.82666667	0.8946609	375
32	0.78451883	1	0.87924971	375
33	0.99197861	0.98933333	0.99065421	375
34	0.8677686	0.84	0.85365854	375
35	0.9816273	0.99733333	0.98941799	375
36	0.81939799	0.98	0.89253188	250
37	0.51829268	1	0.68273092	425
38	0.45138889	0.17333333	0.2504817	375
39	0.90463918	0.936	0.92005243	375
40	0.96440129	0.79466667	0.87134503	375
41	1	0.97866667	0.98921833	375
Average	0.8974818	0.87360264	0.87284809	
Accuracy	0.87132493			

ตารางที่ 4.9 คะแนนของการฝึกสอนโมเดลด้วยอัลกอริทึม K-Nearest Neighbors ร่วมกับ Kalman Filter

Class	Precision	Recall	F1-Score	Support
0	1	0.996	0.99799599	250
1	1	0.996	0.99799599	250
2	0.992	0.992	0.992	250
3	1	1	1	250
4	0.98765432	0.96	0.97363083	250
5	0.77380952	0.78	0.77689243	250

ตารางที่ 4.9 คะแนนของการฝึกสอนโมเดลด้วยอัลกอริทึม K-Nearest Neighbors ร่วมกับ Kalman Filter (ต่อ)

Class	Precision	Recall	F1-Score	Support
6	1	1	1	250
7	1	0.996	0.99799599	250
8	0.99601594	1	0.99800399	250
9	0.99206349	1	0.99601594	250
10	0.776	0.776	0.776	250
11	0.91698113	0.972	0.94368932	250
12	0.98412698	0.992	0.98804781	250
13	0.90127971	0.986	0.9417383	500
14	0.96907217	0.752	0.84684685	250
15	1	1	1	250
16	0.97260274	0.852	0.90831557	250
17	0.91005291	0.91733333	0.91367862	375
18	0.9859944	0.704	0.82147025	500
19	0.64229765	0.984	0.77725119	250
20	0.99447514	0.72	0.83526682	250
21	0.67479675	0.996	0.80452343	250
22	0.9496124	0.98	0.96456693	250
23	0.90804598	0.632	0.74528302	250
24	0.91262136	0.752	0.8245614	250
25	0.87142857	0.976	0.92075472	375
26	0.98965517	0.97288136	0.98119658	295
27	0.86946387	0.99466667	0.9278607	375
28	0.99457995	0.97866667	0.98655914	375
29	0.64874552	0.48266667	0.55351682	375
30	0.97619048	0.984	0.98007968	375
31	0.97118156	0.89866667	0.93351801	375
32	0.81978022	0.99466667	0.89879518	375

ตารางที่ 4.9 คะแนนของการฝึกสอนโมเดลด้วยอัลกอริทึม K-Nearest Neighbors ร่วมกับ Kalman Filter (ต่อ)

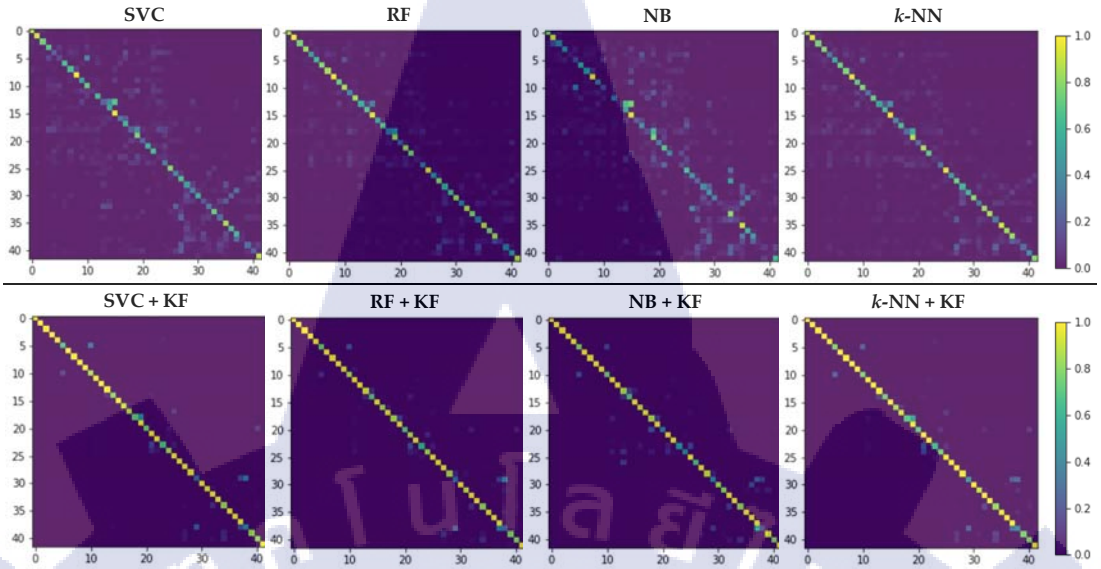
Class	Precision	Recall	F1-Score	Support
33	0.99195711	0.98666667	0.98930481	375
34	0.78826531	0.824	0.80573664	375
35	0.99168975	0.95466667	0.97282609	375
36	0.98	0.98	0.98	250
37	0.65184049	1	0.78922934	425
38	0.52982456	0.40266667	0.45757576	375
39	0.94462541	0.77333333	0.85043988	375
40	0.82825485	0.79733333	0.8125	375
41	0.99457995	0.97866667	0.98655914	375
Average	0.90148244	0.89344821	0.89238307	
Accuracy	0.89255441			

4.4 เปรียบเทียบผลลัพธ์และสรุปผล

จากผลลัพธ์ของโมเดลที่ไม่ประมวลผลข้อมูลด้วยอัลกอริทึม Kalman Filter (ตารางที่ 4.2-4.5) และประมวลผลข้อมูลด้วยอัลกอริทึม Kalman Filter (ตารางที่ 4.6-4.9) จะพบว่าการประมวลข้อมูลด้วยอัลกอริทึม Kalman Filter มีค่าความแม่นยำที่สูงกว่า ซึ่งเมื่อเปรียบเทียบค่าเฉลี่ยพบว่ามีค่าความแม่นยำมากกว่าถึง 35.06% (ตามตารางที่ 4.10) และเปรียบเทียบด้วยตาราง Confusion Matrix พบว่ามีค่าในแนวทแยงเข้าใกล้ 1 มากกว่าอีกด้วย (ตามรูปที่ 4.2)

ตารางที่ 4.10 เปรียบเทียบผลลัพธ์จากการทดลองระหว่างโมเดลทั่วไปกับโมเดลที่ใช้งานร่วมกับ Kalman Filter

ML Models		Accuracy	Precision	Recall	F_1-Score
Traditional	Support Vector Classification (SVC)	0.49744	0.48510	0.51133	0.48394
	Random Forest (RF)	0.65914	0.65745	0.66950	0.65559
	Gaussian Naïve Bayes (NB)	0.38349	0.36350	0.40139	0.35719
	K-nearest Neighbors (k-NN) (k=8)	0.61359	0.60230	0.62263	0.60245
Proposed	SVC + KF	0.90217	0.91226	0.90805	0.90590
	RF + KF	0.89095	0.90364	0.89857	0.89755
	NB + KF	0.87132	0.89748	0.87360	0.87284
	k-NN + KF (k=8)	0.89255	0.90670	0.89797	0.89638
Average Enhancement		35.06%	37.43%	34.33%	36.83%



รูปที่ 4.2 Confusion Matrix ของจุดพื้นที่ทั้ง 42 จุด ผลลัพธ์จากการเรียนรู้ด้วยเครื่องแบบทั่วไปถูกแสดงให้เห็นที่แถวบน และผลลัพธ์จากการเรียนรู้ด้วยเครื่องร่วมกับอัลกอริทึม Kalman Filter ถูกแสดงให้เห็นที่แถวล่าง

บทที่ 5
บทสรุป อภิปราย และข้อเสนอแนะ

การศึกษาวิจัยเรื่อง ระบบระบุตำแหน่งภายในอาคาร โดยใช้วิธีการเรียนรู้เครื่องจักรร่วมกับอัลกอริทึม Kalman Filter ซึ่งการวิจัยครั้งนี้มีวัตถุประสงค์ดังที่กล่าวไปในบทที่ 1 ได้นำเสนอข้อสรุปดังนี้

5.1 สรุปผลการวิจัย

ในการสรุปผลของงานวิจัยชิ้นนี้ ผู้วิจัยได้ทำการสรุปผลตามวัตถุประสงค์ในแต่ละหัวข้อที่ได้กล่าวไปแล้วในบทที่ 1 ดังต่อไปนี้

ระบบระบุตำแหน่งภายในอาคาร โดยใช้วิธีการเรียนรู้เครื่องจักรร่วมกับอัลกอริทึม Kalman Filter นั้นสามารถเก็บข้อมูลค่าความแรงของสัญญาณจากตัวกระจายสัญญาณ Ruuvi ทั้ง 5 ตัวได้ ด้วยอุปกรณ์ฝังตัว Raspberry Pi 4 และสามารถเก็บรวบรวมข้อมูลในรูปแบบไฟล์ csv ได้ด้วย Script ภาษา Python อีกทั้งยังสามารถนำข้อมูลที่ได้ไปใช้ในการฝึกสอนโมเดลได้ และสามารถนำโมเดลที่ได้จากการฝึกสอนมาใช้ในการทำนายตำแหน่งของอุปกรณ์ฝังตัวได้ โดยชุดข้อมูลที่ใช้ในการฝึกสอนอยู่ที่ประมาณไม่น้อยกว่า 750 ตัวอย่างต่อตำแหน่งซึ่งรวมแล้วมีจำนวนตัวอย่างทั้งหมดมากกว่า 31,500 ตัวอย่าง และข้อมูลที่ใช้ในการทดสอบอยู่ที่ประมาณไม่น้อยกว่า 250 ตัวอย่างต่อตำแหน่ง ซึ่งรวมแล้วมีจำนวนตัวอย่างทั้งหมดมากกว่า 10,500 ตัวอย่าง

จากการทดสอบระบบโมเดลที่ถูกฝึกสอนด้วยชุดข้อมูลที่ผ่านการกรองด้วย Kalman Filter และ ชุดข้อมูลที่ไม่ผ่านการกรองพบว่าโมเดลที่ผ่านการกรองด้วย Kalman Filter ทั้งชุดข้อมูลที่ฝึกสอนและทดสอบมีค่าความแม่นยำที่สูงกว่าแบบไม่ผ่านการกรอง

5.2 ปัญหาและอุปสรรคในการทำวิจัย

5.2.1 จากขั้นตอนการเก็บข้อมูลเนื่องจากการรอรับสัญญาณจากตัวกระจายสัญญาณพร้อมกันจำเป็นต้องใช้เวลาในการรอต่อชุดอยู่ที่ 10 วินาทีเพื่อให้ได้รับสัญญาณจากทุกจุดทำให้เวลาที่ใช้ในการเก็บข้อมูลมีระยะเวลาที่นาน

5.2.2 ในส่วนของ Kalman Filter จำเป็นต้องใช้ชุดข้อมูลจำนวนมากเพื่อจะทำให้การกรองข้อมูลมีความแม่นยำ ทำให้ไม่สามารถทำนายผลได้ทันที โดยที่การจะทำให้ได้รับชุดข้อมูลที่เพียงพอจำเป็นต้องใช้เวลาในการเก็บนาน

5.3 ข้อเสนอแนะงานวิจัย

5.3.1 สำหรับงานวิจัยนี้สามารถปรับเปลี่ยนวิธีการกรองข้อมูลได้เพื่อเพิ่มประสิทธิภาพด้านระยะเวลาในการใช้งาน

5.3.2 สำหรับด้านโมเดลของงานวิจัยนี้ยังไม่ได้มีการนำ Neural Network เข้ามาใช้ในการฝึกสอนจึงเป็นจุดที่สามารถนำไปพัฒนาต่อได้ด้วยการนำเทคนิคต่างๆ ทางด้าน Neural Network มาใช้เพื่อเพิ่มประสิทธิภาพให้มีความแม่นยำที่มากกว่าได้โดยไม่ผ่านตัวกรอง

บรรณานุกรม

บรรณานุกรม

[1] V. F. Diggelen and E. Per, "The world's first Gps mooc and worldwide laboratory using Smartphones," *Proceedings of the 2004 Northeast Decision Science Institute Conference, NEDSI 2015 Proceedings,* New Jersey, USA, March 6-9, 2015, pp. 361-369.

[2] A. Nessa et al., "A survey of machine learning for indoor positioning," *KKU Engineering Journal,* vol. 8, no. 3, pp. 267-274, May 2020.

[3] A. Blakeston, *"Understanding RSSI,"* [Online]. Available : https://www.metageek.com/training/resources/understanding-rssi. [Accessed : September 4, 2021].

[4] K. Geok et al., *"Review of Indoor Positioning : Radio Wave Technology,"* Available : https://www.cyber.gov.au/acsc/view-all-content/guidance/tax-scam-stories/rajivs-story. [Accessed : September 4, 2022].

[5] R. Gandhi, *"Support Vector Machine - Introduction to Machine Learning Algorithms,"* [Online]. Available : https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47. [Accessed : June 7, 2018].

[6] S. Sreenivasa, *"Radial Basis Function (RBF) Kernel : The Go-To Kernel,"* [Online]. Available : https://towardsdatascience.com/radial-basis-function-rbf-kernel-the-go-to-kernel-acf0d22c798a. [Accessed : October 12, 2020].

[7] T. Yiu, *"Understanding Random Forest,"* [Online]. Available : https://towards datascience.com/understanding-random-forest-58381e0602d2. [Accessed : June 12, 2019].

[8] W. Ansari, *"Understanding the Maths Behind the Gini Impurity Method for Decision Tree Split,"* [Online]. Available : https:// analyticsindiamag.com/under standing-the-maths-behind-the-gini-impurity-method-for-decision-tree-split/#:~:text=Gini%20impurity%20is%20an%20important%20measure%20used%. [Accessed : March 18, 2022].

[9] R. Vats, *"Gaussian Naive Bayes : What You Need to Know?,"* [Online]. Available : https://www.upgrad.com/blog/gaussian-naive-bayes/#:~:text=Gaussian%20Na%C3%AFve%20Bayes%20is%20the,deviation%20for%20the%20training%20data. [Accessed : February 22, 2021].

[10] S. J. Bigelow, *"K-Nearest Neighbor (KNN) Algorithm for Machine Learning,"* [Online]. Availble : https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning. [Accessed : March 18, 2022].

[11] W. Bulten et al., "Human slam, indoor localisation of devices and users," *IEEE International Conference on Recent Advances in Information Technology,* RAIT, Berlin, Germany, March 3-5, 2016, pp. 211-222.

[12] P. Sthapit et al., "Bluetooth based indoor positioning using machine learning algorithms," *IEEE International Conference on Consumer Electronics-Asia,* IACC 2018, Bhimavaram, India, February 27-28, 2018, pp. 206-212.

[13] A. Sashida et al., "A machine learning approach to indoor positioning for mobile targets using signals," *International Conference on Intelligence Computation and Applications, ISICA 2019 Proceedings,* JeJu, Korea (South), March 2019, pp. 1-4.

[14] V. Stavrou et al., "An ensemble filter for indoor positioning in a retail store using bluetooth low energy beacons," *Journal of Hydrology*, vol. 358, no. 3, pp. 294-303, September 2019.