รายงาน

เรื่อง Cardiovascular Disease prediction

เสนอ

อาจารย์ รศ.ดร. พีรพล ศิริพงศ์วุฒิกร

อาจารย์ ผศ.ดร. สันติธรรม พรหมอ่อน

อาจารย์ ดร. อัญชลิสา แต้ตระกูล

จัดทำโดย

นางสาวปภิญญ์ ธนัชพัทธ์ รหัสนักศึกษา 62070505209

นางสาวปิยธิดา มีเสถียร รหัสนักศึกษา 62070505210

นางสาวจิราพัชร พรเจริญวิโรจน์ รหัสนักศึกษา 62070505235

นางสาวอรกร เมฆชัยพร รหัสนักศึกษา 62070505238

รายงานฉบับนี้เป็นส่วนหนึ่งของรายวิชา CPE 378 การเรียนรู้ของเครื่อง (Machine Learning)

สาขาวิชาวิทยาศาสตร์ข้อมูลสุขภาพ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าธนบุรี

ภาคเรียนที่ 2 ปีการศึกษา 2564

คำนำ

รายงานฉบับนี้จัดทำขึ้นเพื่อเป็นส่วนหนึ่งของวิชา CPE 378 การเรียนรู้ของเครื่อง (Machine Learning) จัดทำขึ้นเพื่อศึกษาอัลกอริทึมที่ใช้ในการวินิจฉัยโรคหัวใจและหลอดเลือด (cardiovascular disease, CVD) ซึ่งเป็นโรคที่มีสถิติการเสียชีวิตสูง โดยในรายงานฉบับนี้มีอัลกอริทึมที่นำมาใช้ในการวินิจฉัยโรคหัวใจและหลอดเลือดทั้งหมด 3 แบบ คือ Logistic Regression Classifier, Random Forest Classifier, และ eXtreme Gradient Boosting (XGBoost) ใช้ในการสร้างแบบจำลองที่สามารถนำมาวิเคราะห์และวินิจฉัยโรคจากข้อมูล Cardiovascular Disease prediction และเปรียบเทียบหาอัลกอรึทึมที่สามารถสร้างแบบจำลองมาวินิจฉัยโรคได้ถูกต้องที่สุด

คณะผู้จัดทำคาดหวังเป็นอย่างยิ่งว่าการจัดทำรายงานฉบับนี้จะมีข้อมูลที่เป็นประโยชน์ต่อผู้ที่สนใจศึกษาพื้นฐานของการสำรวจข้อมูล การเตรียมข้อมูล และสนใจศึกษาพื้นฐานอัลกอริทึม Logistic Regression Classifier, Random Forest Classifier, และ eXtreme Gradient Boosting (XGBoost) เพื่อใช้ในการแบ่งกลุ่ม

คณะผู้จัดทำ

สารบัญ

Cardiovascular Disease prediction

บทที่ 1

บทนำ

ที่มาและความสำคัญ

โรคหัวใจและหลอดเลือด (cardiovascular disease, CVD) เป็นกลุ่มโรคที่เกิดจากความผิดปกติของหัวใจหรือหลอดเลือด ได้แก่ โรคหลอดเลือดหัวใจ (coronary heart disease) โรคหลอดเลือดสมอง (cerebrovascular disease) โรคเส้นเลือดแดงส่วนปลายอุดตัน (peripheral artery disease) ภาวะหลอดเลือดดำอุดตัน (deep vein thrombosis) และภาวะลิ่มเลือดอุดตันในปอด (pulmonary embolism) เป็นต้น ปัจจัยเสี่ยงหลักที่สำคัญของโรคหัวใจและหลอดเลือด เกิดจากพฤติกรรมการใช้ชีวิตประจำวันที่ไม่ถูกต้อง ได้แก่ การรับประทานอาหารที่มีรสเค็มจัด หรืออาหารไขมันสูง การไม่ออกกำลังกาย การสูบบุหรี่ และการดื่มแอลกอฮอล์ ปัจจัยเสี่ยงดังกล่าวจะนำไปสู่ภาวะความดันโลหิตสูง ระดับน้ำตาลในเลือดสูง ระดับไขมันในเลือดสูง ภาวะน้ำหนักเกินหรือโรคอ้วน และเกิดโรคหัวใจและหลอดเลือดในที่สุด โรคหัวใจและหลอดเลือดเป็นโรคที่พบมากในผู้สูงอายุ[1]

จากการรายงานสถิติขององค์การอนามัยโลก (WHO) ในปี 2563 พบว่า โรคหัวใจและหลอดเลือด (cardiovascular diseases) คือกลุ่มโรคที่เป็นสาเหตุการตายอันดับ 1 ของคนทั่วโลก โดยมีผู้เสียชีวิตจากกลุ่มโรคนี้ประมาณ 17.9 ล้านคน และสำหรับสถิติข้อมูลการเสียชีวิตของคนไทย ในกลุ่มโรคหัวใจและหลอดเลือด พบว่า ร้อยละ 80 เสียชีวิตด้วยกล้ามเนื้อหัวใจขาดเลือดเฉียบพลัน อีกทั้งข้อมูลจากกรมการแพทย์ ปี 2557 พบว่า ประเทศไทยมีค่าใช้จ่ายในการรักษาพยาบาลเฉลี่ยของผู้ป่วยโรคหัวใจถึง 6,906 ล้านบาทต่อปี และมีแนวโน้มการป่วยเพิ่มขึ้นอย่างต่อเนื่อง ซึ่งส่งผลกระทบต่อคุณภาพชีวิตของประชาชน เกิดความสูญเสียทางเศรษฐกิจจากการเสียชีวิตก่อนวัยอันควร ทั้งในระดับบุคคล ครอบครัว สังคม และประเทศชาติ[2]

จากสถิติที่กล่าวว่าโรคโรคหัวใจและหลอดเลือดเป็นสาเหตุสำคัญในการเสียชีวิตของประชากรส่วนใหญ่ ซึ่งวิธีในการช่วยเพิ่มโอกาสในการรักษาและเพิ่มอัตราการรอดชีวิตสามารถทำได้หากวิฉัยโรคโรคได้เร็ว การทำเทคโนโลยีทางคอมพิวเตอร์อย่างการเรียนรู้ของเครื่องเข้ามาช่วยในการวินิจฉัยจึงเป็นหนึ่งในทางเลือกที่เหมาะสม

ดังนั้นในการทำ Project ในครั้งนี้ คณะผู้จัดทำจึงเลือกอัลกอริทึม 3 แบบ คือ Logistic Regression Classifier, Random Forest Classifier, และ eXtreme Gradient Boosting (XGBoost) นำมาใช้ในการสร้างแบบจำลองทำนายความเสี่ยงโรคหลอดเลือดหัวใจจากชุดข้อมูล Cardiovascular Disease โดยมีวัตถุประสงค์เพื่อเปรียบเทียบความถูกต้องของอัลกอริทึมทั้ง 3 แบบ เพื่อหาอัลกอริทึมที่สามารถสร้างแบบจำลองในการทำนายความเสี่ยงโรคได้ถูกต้องที่สุด

บทที่ 2
การวิเคราะห์อัลกอริทึมที่ใช้เบื้องต้น

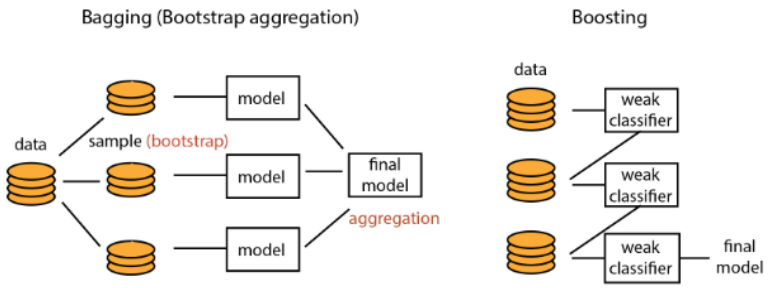
Ensemble Method

เป็นวิธีการปรับปรุงแบบจำลองเดิมให้สามารถรับมือกับปัญหาที่มีความซับซ้อนมากขึ้นได้ ซึ่งเป็นการรวบรวมหลายแบบจำลองด้วยวิธีต่าง ๆ เพื่อให้ได้แบบจำลองสุดท้ายที่มีประสิทธิภาพและแม่นยำมากขึ้น

Ensemble ใน Machine Learning ที่มักใช้กันบ่อยมี 2 วิธี ได้แก่

1. Bagging (ย่อมาจาก Bootstrap Aggregation) ซึ่งเป็นพื้นฐานของ **Random Forest Classifier** ใน scikit-learn library

2. Boosting ซึ่งเป็นพื้นฐานของ AdaBoost หรือ Gradient Boosting ในไลบรารี่เช่น **XGBoost** และ LightGBM



รูปที่ 1 ความแตกต่างระหว่าง Bagging และ Boosting (Machine Learning)
(ที่มา [ML] Bagging หรือ Boosting คืออะไร ทำงานอย่างไร? – tupleblog)

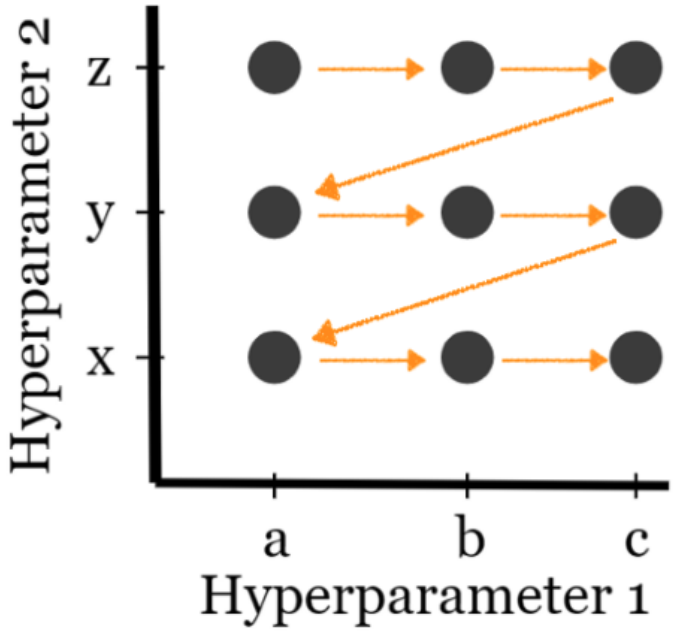
Bagging (Bootstrap Aggregation)

คือการสุ่มตัวอย่างข้อมูลจากชุดข้อมูลทั้งหมดแล้วสร้างแบบจำลองขึ้นมา สำหรับวิธีการสุ่มข้อมูลออกมา จะใช้วิธีสุ่มแบบแทนที่ (random with replacement) ข้อมูลที่มีจะไม่ได้ลดลงหลังจากการสุ่ม สามารถสุ่มข้อมูลหลาย ๆ รอบเพื่อให้ได้แบบจำลองหลาย ๆ ตัว แล้วนำข้อมูลย่อย ๆ มาฝึกแบบจำลองด้วยวิธีเดียวกันพร้อม ๆ กัน (Pararell Method) โดยหลังจากการฝึกจะนำแบบจำลองทั้งหมดมารวมกันด้วยวิธีหาค่าเฉลี่ย (Averaging) หรือการให้น้ำหนักค่าที่พบมากที่สุด (Voting) วิธีการสุ่มข้อมูลนี้ทำให้ได้แบบจำลองหลาย ๆ ตัวมาช่วยกันทำนาย ซึ่งสามารถลดแนวโน้มที่แบบจำลองจะจำเพาะต่อชุดข้อมูลฝึก (overfit)

Boosting

คือการนำ weak classifier หรือ classifier ที่มีความแม่นยำต่ำมา ทำนายข้อมูลที่มี จากนั้นจะให้ weak classifier ตัวใหม่มาแก้ไข error ที่มี โดยผลรวมของ classifier จะเกิดเป็น classifier ใหม่ขึ้นมา ทำซ้ำจนได้แบบจำลองที่ดีที่สุดจากผลรวมของ classifier ซึ่งสามารถนำมาทำนายข้อมูลที่ซับซ้อนมากๆได้ ข้อเสียของการใช้ boosting ก็คือต้องรันหลาย ๆ ครั้งและเป็นลำดับกว่าจะได้แบบจำลองที่ต้องการ ต่างจาก Bagging ที่สามารถสุ่มข้อมูลได้แล้วฝึกแบบจำลองได้พร้อม ๆ กัน[3]

GridSearchCV



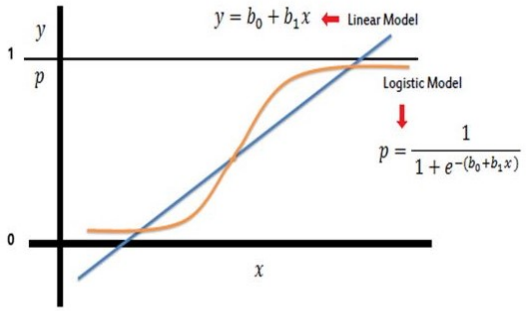
รูปที่ 2 Grid Search

(ที่มา A Practical Introduction to Grid Search, Random Search, and Bayes Search)

เป็นไลบรารี่ sklearn สำหรับการปรับพารามิเตอร์ในโมเดล GridSearchCV เป็นอัลกอริทึมประเภท Exhaustive search ที่จะทำการค้นหาพารามิเตอร์ที่เหมาะสมสำหรับโมเดลตามค่าของพารามิเตอร์ที่กำหนดใน parameter grid ซึ่งเป็น attribute หนึ่งใน GridSearchCV โดย parameter grid เป็นตัวแปรที่ประเภท dictionary เก็บค่า parameter เป็น key และเก็บ list ค่าพารามิเตอร์ เป็น value ผู้ใช้จำเป็นจะต้องมีการกำหนดค่าของพารามิเตอร์ที่สนใจเป็น list โดยพารามิเตอร์จะถูกเพิ่มประสิทธิภาพด้วยการทำ cross-validated grid-search เพื่อลดความจำเพาะต่อชุดข้อมูลที่ใช้ฝึกโมเดล[4]

1. Logistic Regression Classifier

Logistic regression เป็น classifier algorithm เป็นเทคนิคการวิเคราะห์สถิติเชิงคุณภาพ นิยมใช้มากในการวิจัยทางด้านการแพทย์และสาธารณสุข[5] เป็นเทคนิคการวิเคราะห์ความสัมพันธ์ของตัวแปรที่มีเป้าหมายการวิเคราะห์คือ ทำนายหรือคาดการณ์โอกาสของการเกิดขึ้น และไม่เกิดขึ้นที่เป็นไปได้ของข้อมูลที่สนใจ[6]



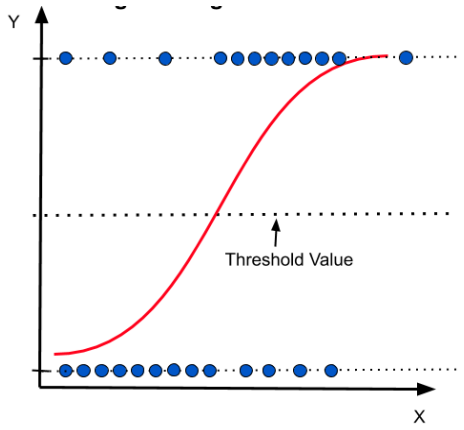
รูปที่ 3 Understanding Logistic Regression

(ที่มา Step by Step explanation of Logistics Regression - From The GENESIS)

โดย Logistic Regression สามารถแบ่งได้เป็น 2 ประเภท คือ Binary Logistic Regression Analysis และ Multinomial Logistic Regression Analysis ใช้สำหรับการจำแนกหลายกลุ่มอาจแสดงได้ด้วยกราฟมากกว่า 1 กราฟ ซึ่งการทำ Project นี้เราใช้ Binary Logistic Regression Analysis ซึ่งเป็นการวิเคราะห์ที่ตัวแปรตามมี 2 ค่า คือ ไม่เกิดขึ้น(0) หรือเกิดขึ้น(1) เพื่อมาใช้ในการพยากรณ์โอกาสของการเป็นและไม่เป็นโรคหัวใจและหลอดเลือด[7]

โดยมีสถิติเบื้องหลังความสัมพันธ์ระหว่างตัวแปรทำนาย กับตัวแปรเกณฑ์ของ Logistic Regression ที่ไม่เป็นรูปเชิงเส้น และต้องมีการปรับความสัมพันธ์ให้อยู่ในรูปเชิงเส้น ในรูปแบบของ Odds หรือ Odd Ratio ซึ่งเป็นอัตราส่วนระหว่างโอกาสที่จะเกิดเหตุการณ์ที่สนใจ P_y และโอกาสที่จะไม่เกิดเหตุการณ์ที่สนใจ 1 - P_y

ค่าของ Odds จะแสดงถึงโอกาสที่จะเกิดเหตุการณ์ที่สนใจ เป็นกี่เท่าของโอกาสที่จะไม่เกิดเหตุการณ์ที่สนใจมีค่าระหว่าง -∞ ถึง ∞ และหลังจากการนำ Odds ที่ได้ไปหาความน่าจะเป็นจะได้ค่าจากการแปลงระหว่าง 0 ถึง 1 โดยค่าที่เข้าใกล้ -∞ จะมีค่าเข้าใกล้ 0 และค่าที่เข้าใกล้ ∞ จะมีค่าเข้าใกล้ 1 ซึ่งจะแบ่งกลุ่มโดยใช้ค่า Threshold Value เป็นเกณฑ์



รูปที่ 4 Logistic Regression

(ที่มา Why Is Logistic Regression Called “Regression” If It Is A Classification Algorithm?)

การเขียน Logistic Model จะอยู่ในรูปของ log ของ odds เรียกว่า logit

$$\text{Log } P_y / (1 - P_y) = w_0 + w_1 x_1 + \ldots + w_p x_p$$

โดยกำหนดให้ค่า

P_y = ความน่าจะเป็นของการเกิดเหตุการณ์ y ที่สนใจ

w = น้ำหนักของแต่ละตัวแปรของข้อมูล

X = ตัวแปรของข้อมูล

และ สำหรับการทำนายค่า y ที่เป็น Py ในการ Logistic regression จะใช้สมการ

$$P_y = e^{w_0 + w_1 x_1 + \ldots + w_p x_p} / 1 + e^{w_0 + w_1 x_1 + \ldots + w_p x_p}$$

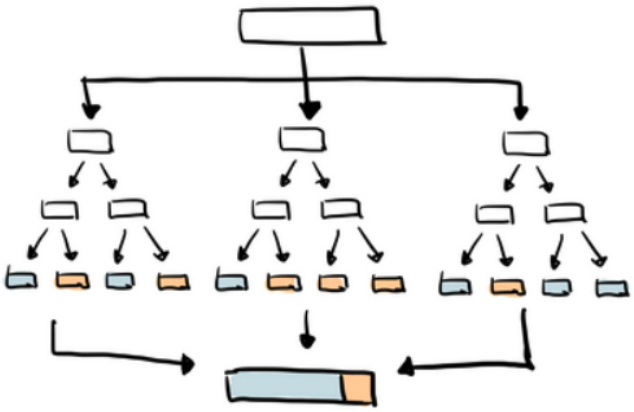
ตามวิธี maximum likelihood ในขณะที่การทำนายค่า y ในการวิเคราะห์การถดถอยปกติจะใช้วิธี least square จากสมการ $y = w_0 + w_1 x_1 + \ldots + w_p x_p$ [8]

2. Random Forest Classifier

เป็น **Ensemble model** ประเภทหนึ่งที่มีพื้นฐานมาจากต้นไม้ตัดสินใจ (Decision tree) ซึ่งเป็นแบบจำลองที่อ้างอิงตามกฎ (Rule-based) ที่จะสร้างกฎ If-else condition โดยจำลองว่าถ้าเกิดเหตุการณ์ A แล้วผลลัพธ์จะเป็นอย่างไร และหากไม่เกิดเหตุการณ์ A จะได้ผลลัพธ์อย่างไร โดยแบบจำลอง Decision tree จะทำแบบนี้กับทุกๆตัวแปรที่มี แต่ปัญหาที่เกิดขึ้นจากต้นไม้ตัดสินใจคือ เป็นแบบจำลองที่มีความอ่อนไหวต่อข้อมูลสูงมาก กล่าวเพิ่มเติมคือ หากเพิ่มเติมตัวแปรใหม่มา แบบจำลองจะต้องคำนวณและจัดลำดับใหม่ทั้งหมด ทำให้มีโอกาสเกิดเหตุการณ์ที่แบบจำลองสามารถทำนายผลจากข้อมูลที่ใช้ฝึกได้ดีแต่ไม่สามารถทำนายผลจากข้อมูลใหม่ๆได้แม่นยำนัก (Overfitting) ทำให้เกิด Random Forest ขึ้นมาเพื่อแก้ปัญหาดังกล่าว โดยแนวคิดคือการสร้างต้นไม้หลายต้น ที่คัดเอาบางส่วนของทั้งข้อมูลและตัวแปร มาใช้ในการฝึกแบบจำลอง เมื่อแบบจำลองถูกฝึกมาตามจำนวนที่กำหนด (โดยปกติจะถูกตั้งค่าเริ่มต้นไว้ที่ 100 ต้นใน library scikit-learn) จะนำมารวมกันเป็นแบบจำลองเดียว ซึ่งข้อดีของ Random Forest คือ ประสิทธิภาพความแม่นยำของแบบจำลองมากกว่า Decision tree, สามารถทำงานได้ทั้งงานประเภท classification และ regression, เนื่องจากเป็นแบบจำลองที่พัฒนาจากแบบจำลอง rule-based ทำให้ไม่จำเป็นต้องมีสมมติฐานของข้อมูลว่าต้องมีการกระจายตัวแบบปกติหรือ ต้องมีความสัมพันธ์แบบเป็นเส้นตรง แต่ข้อเสียคือไม่สามารถอธิบายแต่ละตัวแปรแบบ if else condition ได้ (เนื่องจากมีการรวมกันของหลายแบบจำลองและมีการปรับรูปแบบไปแล้ว เช่น ใช้ค่าเฉลี่ย หรือใช้การให้น้ำหนักค่าที่พบมาก) และต้องมีชุดข้อมูลที่มีขนาดใหญ่

หลักการของ Random Forest

คือ สร้างแบบจำลองจาก Decision Tree ขึ้นมาหลายแบบจำลอง ประมาณ 500-1000 ที่ไม่ซ้ำกัน แล้วฝึกแบบจำลองที่เหมือนกันหลาย ๆ ครั้ง บนข้อมูลชุดเดียวกัน โดยแต่ละครั้งที่ฝึก แต่ละแบบจำลองจะได้รับข้อมูลไม่เหมือนกัน ซึ่งเป็นข้อมูลย่อยของข้อมูลทั้งหมด โดยใช้วิธีการสุ่มหรือเทคนิค Bagging[9]



รูปที่ 5 อัลกอริทึม Random Forest

(ที่มา Top 6 Machine Learning Algorithms for Classification)

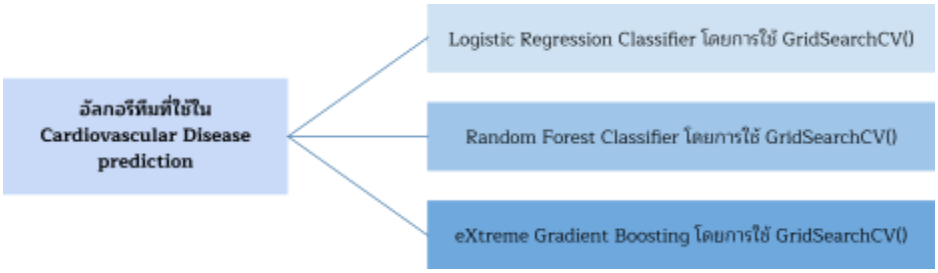
กรณี classification ที่นำ Random Forest มาใช้ ในการทำการพยากรณ์ (prediction) ของแต่ละ Decision Tree จะทำการพยากรณ์แยกกัน และคำนวณผลการพยากรณ์สุดท้ายจากผลการพยากรณ์ที่ถูกเลือกโดย Decision Tree มากที่สุด (Voting)[10]

3. eXtreme Gradient Boosting (XGBoost)

แบบจำลองที่ถูกพัฒนาขึ้นมาต่อจาก Gradient boosting เพื่อเพิ่มความแม่นยำและความยืดหยุ่นให้กับแบบจำลอง โดยใช้หลักการของ **Ensemble Learning Method** โดยพัฒนาขึ้นเพื่อให้ทำงานได้เร็วยิ่งขึ้นกว่าเดิมในการค้นหาตัวแปรที่จะส่งผลต่อแบบจำลองมากที่สุดแทนการใช้ทุก ๆ ตัวแปรแบบ gradient boosting ดั้งเดิม ในการ Boosting เพื่อสร้างตัวเรียนรู้หลายๆตัว (Multiple Learner)[11] หรือเรียกได้ว่าเป็นการรวม Weak Learners หลาย ๆ ตัวมาทำงานเป็นโซ่ต่อเข้าด้วยกัน ซึ่ง Learner ที่สร้างขึ้นใหม่แต่ละรุ่นนั้นจะทำการแก้ไขข้อบกพร่องในการทำงานของ Learner รุ่นก่อนหน้าเพื่อลด Error พอฝึกแบบจำลองเสร็จแล้ว Learner ทุกตัวจะพยากรณ์ร่วมกัน[12]

บทที่ 3
ขั้นตอนการออกแบบการวิเคราะห์

การนำอัลกอริทึมทั้ง 3 แบบ ไปใช้ใน Cardiovascular Disease prediction



รูปที่ 6 แสดงการลำดับการนำอัลกอริทึมทั้ง 3 ไปใช้ใน Cardiovascular Disease prediction

การนำอัลกอริทึมไปใช้งานใน Cardiovascular Disease prediction จะแบ่งออกเป็นดังนี้

1. Logistic Regression โดยการใช้ GridSearchCV()
2. Random Forest โดยการใช้ GridSearchCV()
3. eXtreme Gradient Boosting โดยการใช้ GridSearchCV()

บทที่ 4
การวิเคราะห์ และจัดการข้อมูลเบื้องต้น

ข้อมูลที่ใช้ในการวิเคราะห์ ประกอบด้วย ดังนี้ Cardiovascular Disease Dataset และ Synthetic Dataset

1. Cardiovascular Disease Dataset [13]

Cardiovascular Disease Dataset เป็นข้อมูลเกี่ยวกับโรคหัวใจและหลอดเลือดของคนไข้ทั้ง 70,000 คน โดยมีผู้ป่วยเป็นโรคหัวใจและหลอดเลือด 34,979 คน และผู้ป่วยที่ไม่เป็นโรคหัวใจและหลอดเลือด 35,021 คน สามารถเข้าถึงได้โดย Cardiovascular Disease dataset | Kaggle

```
In [1]:
import pandas as pd
cardio_df = pd.read_csv(r'cardio_train.csv', sep=';')
cardio_df = cardio_df.drop(columns='id')
cardio_df.head()
```

Out[1]:

	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
0	18393	2	168	62.0	110	80	1	1	0	0	1	0
1	20228	1	156	85.0	140	90	3	1	0	0	1	1
2	18857	1	165	64.0	130	70	3	1	0	0	0	1
3	17623	2	169	82.0	150	100	1	1	0	0	1	1
4	17474	1	156	56.0	100	60	1	1	0	0	0	0

รูปที่ 7 ตัวอย่างข้อมูล 5 แถวแรกของ Cardiovascular Disease Dataset

จาก Cardiovascular Disease Dataset มีทั้งหมด 70,000 datapoints โดยประกอบด้วย 12 Columns ดังนี้

1. **age** คือ อายุ มีหน่วยเป็น วัน (days)
2. **gender** คือ เพศ โดย 1 หมายถึง เพศหญิง และ 2 หมายถึง เพศชาย
3. **height** คือ ส่วนสูง มีหน่วยเป็น เซนติเมตร (cm)
4. **weight** คือ น้ำหนัก มีหน่วยเป็น กิโลกรัม (kg)
5. **ap_hi** คือ ความดันช่วงหัวใจบีบตัว (Systolic blood pressure, SBP) หรือความดันตัวบน มีหน่วยเป็นมิลลิเมตรปรอท (mmHg) ค่าปกติคือ 90-140 mmHg
6. **ap_lo** คือ ความดันช่วงหัวใจคลายตัว (Diastolic Blood Pressure, DBP) หรือความดันตัวล่าง มีหน่วยเป็นมิลลิเมตรปรอท (mmHg) ค่าปกติคือ 60-90 mmHg
7. **cholesterol** คือ โคเลสเตอรอล ซึ่งเป็นไขมันที่พบอยู่อยู่ในร่างกาย โดย 1 หมายถึง ปกติ 2 หมายถึง ผิดปกติ และ 3 หมายถึง ผิดปกติมาก

8. **gluc** คือ ระดับน้ำตาลในเลือด โดย 1 หมายถึง ปกติ 2 หมายถึง ผิดปกติ และ 3 หมายถึง ผิดปกติมาก
9. **smoke** คือ การสูบบุหรี่ โดย 0 หมายถึง ไม่สูบบุหรี่ และ 1 หมายถึง สูบบุหรี่
10. **alco** คือ การดื่มแอลกอฮอล์ โดย 0 หมายถึง ไม่ดื่มแอลกอฮอล์ และ 1 หมายถึง ดื่มแอลกอฮอล์
11. **active** คือ การออกกำลังกาย โดย 0 หมายถึง ไม่ออกกำลังกาย และ 1 หมายถึง ออกกำลังกาย
12. **cardio** คือ การป่วยเป็นโรคหลอดเลือดหัวใจ โดย 0 หมายถึง ไม่ป่วยเป็นโรคหลอดเลือดหัวใจ และ 1 หมายถึง ป่วยเป็นโรคหลอดเลือดหัวใจ

1.1 Exploratory Data Analysis (EDA)

- นำเข้าข้อมูล

In [1]:
```
import pandas as pd
cardio_df = pd.read_csv(r'cardio_train.csv', sep=';')
cardio_df = cardio_df.drop(columns='id')
cardio_df.head()
```

Out[1]:

	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
0	18393	2	168	62.0	110	80	1	1	0	0	1	0
1	20228	1	156	85.0	140	90	3	1	0	0	1	1
2	18857	1	165	64.0	130	70	3	1	0	0	0	1
3	17623	2	169	82.0	150	100	1	1	0	0	1	1
4	17474	1	156	56.0	100	60	1	1	0	0	0	0

รูปที่ 8 ตัวอย่างข้อมูล 5 แถวแรกของ Cardiovascular Disease Dataset

- ตรวจสอบข้อมูล

In [9]:
```
cardio_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70000 entries, 0 to 69999
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   age          70000 non-null  int64  
 1   gender       70000 non-null  int64  
 2   height       70000 non-null  int64  
 3   weight       70000 non-null  float64
 4   ap_hi        70000 non-null  int64  
 5   ap_lo        70000 non-null  int64  
 6   cholesterol  70000 non-null  int64  
 7   gluc         70000 non-null  int64  
 8   smoke        70000 non-null  int64  
 9   alco         70000 non-null  int64  
 10  active       70000 non-null  int64  
 11  cardio       70000 non-null  int64  
dtypes: float64(1), int64(11)
memory usage: 6.4 MB
```

รูปที่ 9 ตรวจสอบข้อมูลของ Cardiovascular Disease Dataset

จากการตรวจสอบข้อมูล พบว่า มี 70,000 datapoints และประกอบด้วย 12 Columns โดยเป็น numerical data ทั้งหมด ซึ่งมี 12 integers และ 1 float

- ตรวจสอบ null ในข้อมูล

```
In [3]: 1 cardio_df.isnull().any()

Out[3]: id             False
        age            False
        gender         False
        height         False
        weight         False
        ap_hi          False
        ap_lo          False
        cholesterol    False
        gluc           False
        smoke          False
        alco           False
        active         False
        cardio         False
        dtype: bool
```

รูปที่ 10 ตรวจสอบ null ของ Cardiovascular Disease Dataset

จากการตรวจสอบ null ในข้อมูล พบว่า ในข้อมูลไม่มี Null

- Descriptive statistics

```
In [10]: 1 cardio_df.describe()
```

Out[10]:

	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
count	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000	70000.000000
mean	19468.865814	1.349571	164.359229	74.205690	128.817286	96.630414	1.366871	1.226457	0.088129	0.053771	0.803729	0.499700
std	2467.251667	0.476838	8.210126	14.395757	154.011419	188.472530	0.680250	0.572270	0.283484	0.225568	0.397179	0.500003
min	10798.000000	1.000000	55.000000	10.000000	-150.000000	-70.000000	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000
25%	17664.000000	1.000000	159.000000	65.000000	120.000000	80.000000	1.000000	1.000000	0.000000	0.000000	1.000000	0.000000
50%	19703.000000	1.000000	165.000000	72.000000	120.000000	80.000000	1.000000	1.000000	0.000000	0.000000	1.000000	0.000000
75%	21327.000000	2.000000	170.000000	82.000000	140.000000	90.000000	2.000000	1.000000	0.000000	0.000000	1.000000	1.000000
max	23713.000000	2.000000	250.000000	200.000000	16020.000000	11000.000000	3.000000	3.000000	1.000000	1.000000	1.000000	1.000000

รูปที่ 11 Descriptive statistics ของ Cardiovascular Disease Dataset

จากการตรวจสอบ Descriptive statistics พบว่า มีความผิดปกติตรงค่า Min และ ค่า Max ใน Column ดังนี้

- **height:** ส่วนสูงที่น้อยที่สุดคือ 55 cm และมากที่สุดคือ 250 cm
- **weight:** น้ำหนักที่น้อยที่สุดคือ 10 kg และมากที่สุดคือ 200 kg
- **ap_hi:** ความดันช่วงหัวใจบีบตัวที่น้อยที่สุดคือ -150 mmHg และมากที่สุดคือ 16,020 mmHg ซึ่งค่าความดันช่วงหัวใจบีบตัวจะติดลบไม่ได้ และไม่ควรน้อยกว่า 80 mmHg และมากกว่า 180 mmHg เพราะจะกลายเป็น emergency case
- **ap_lo:** ความดันช่วงหัวใจคลายตัวที่น้อยที่สุดคือ -70 mmHg และมากที่สุดคือ 11,000 ซึ่งค่าความดันช่วงหัวใจคลายตัวจะติดลบไม่ได้ และไม่ควรน้อยกว่า 60 mmHg และมากกว่า 110 mmHg เพราะจะกลายเป็น emergency case

ดังนั้นเพื่อจัดการกับข้อมูลที่มีความผิดปกติดังกล่าว จึงต้องทำ Preparing Data ในขั้นตอนต่อไป

- สรุปการกระจายตัวข้อมูล และสัดส่วนข้อมูลแต่ละ Columns ในข้อมูล

1. age

```
In [25]:
cardio_feature = cardio_df.iloc[:,:-1]
cardio_result = cardio_df.iloc[:,-1]
```

```
In [26]:
cardio_feature["age"] = cardio_feature["age"].apply(lambda x : x/365)
cardio_feature
```

Out[26]:

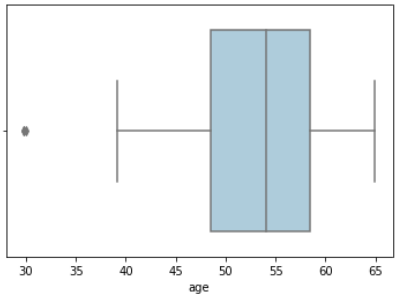
	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active
0	50.391781	2	168	62.0	110	80	1	1	0	0	1
1	55.419178	1	156	85.0	140	90	3	1	0	0	1
2	51.663014	1	165	64.0	130	70	3	1	0	0	0
3	48.282192	2	169	82.0	150	100	1	1	0	0	1
4	47.873973	1	156	56.0	100	60	1	1	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...
69995	52.712329	2	168	76.0	120	80	1	1	1	0	1
69996	61.920548	1	158	126.0	140	90	2	2	0	0	1
69997	52.235616	2	183	105.0	180	90	3	1	0	1	0
69998	61.454795	1	163	72.0	135	80	1	2	0	0	0
69999	56.273973	1	170	72.0	120	80	2	1	0	0	1

70000 rows × 11 columns

รูปที่ 12 ขั้นตอนการเปลี่ยนแปลง age Column ที่มีหน่วยเป็นวัน ให้เป็น ปี

```
In [79]:
import seaborn as sns
sns.boxplot(data = cardio_feature, x = 'age', palette="Paired")
```

```
Out[79]: <AxesSubplot:xlabel='age'>
```

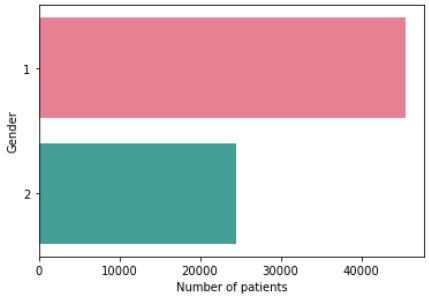


รูปที่ 13 แสดง box plot ของ age Column ที่มีหน่วยเป็นปี

2. gender

```
In [100]:
1 gender_plot = sns.countplot(data = cardio_feature, y = 'gender',palette="husl")
2 gender_plot.set_xlabel("Number of patients")
3 gender_plot.set_ylabel("Gender")
```

```
Out[100]: Text(0, 0.5, 'Gender')
```



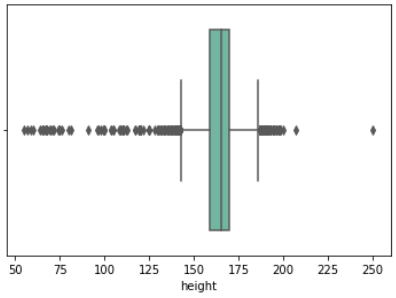
รูปที่ 14 แสดงสัดส่วนของข้อมูล gender Column

จากการแสดงสัดส่วนของข้อมูล gender Column พบว่า มีจำนวนผู้ป่วยเพศหญิง(1) มากกว่า จำนวนผู้ป่วยเพศชาย(2)

3. height

```
In [29]:
1 import seaborn as sns
2 sns.boxplot(data = cardio_df, x = 'height', palette="Set2")
```

```
Out[29]: <AxesSubplot:xlabel='height'>
```



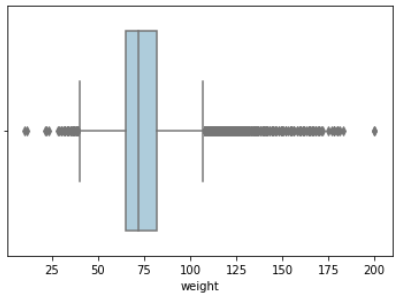
รูปที่ 15 แสดง box plot ของ height Column

จาก box plot ของ height Column พบ Outliers จำนวนมาก จึงต้องทำ Data Preparation ในขั้นตอนต่อไป

4. weight

```
In [30]:
1 import seaborn as sns
2 sns.boxplot(data = cardio_df, x = 'weight', palette="Paired")
```

```
Out[30]: <AxesSubplot:xlabel='weight'>
```

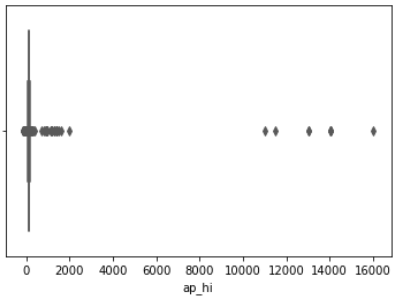


รูปที่ 16 แสดง box plot ของ weight Column

จาก box plot ของ weight Column พบ Outliers จำนวนมาก จึงต้องทำ Data Preparation ในขั้นตอนต่อไป

5. ap_hi

```
In [77]:
1 import seaborn as sns
2 sns.boxplot(data = cardio_df, x = 'ap_hi', palette="Set2")
```



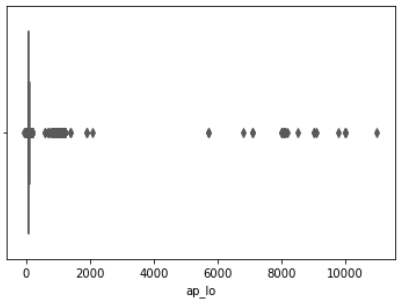
รูปที่ 17 แสดง box plot ของ ap_hi Column

จาก box plot ของ ap_hi Column พบ Outliers จำนวนมาก จึงต้องทำ Data Preparation ในขั้นตอนต่อไป

6. ap_lo

```
In [78]:
1 import seaborn as sns
2 sns.boxplot(data = cardio_df, x = 'ap_lo', palette="Set2")

Out[78]: <AxesSubplot:xlabel='ap_lo'>
```



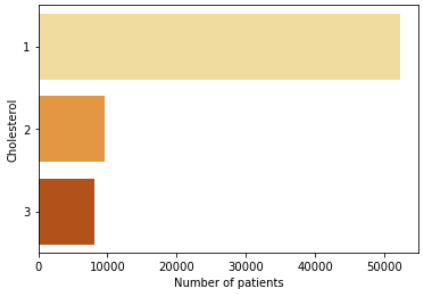
รูปที่ 18 แสดง box plot ของ ap_hi Column

จาก box plot ของ ap_hi Column พบ Outliers จำนวนมาก จึงต้องทำ Data Preparation ในขั้นตอนต่อไป

7. cholesterol

```
In [103]:
1 cholesterol_plot = sns.countplot(data = cardio_feature, y = 'cholesterol',palette="YlOrBr")
2 cholesterol_plot.set_xlabel("Number of patients")
3 cholesterol_plot.set_ylabel("Cholesterol")

Out[103]: Text(0, 0.5, 'Cholesterol')
```



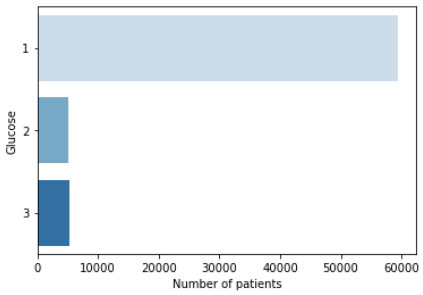
รูปที่ 19 แสดงสัดส่วนของข้อมูล cholesterol Column

จากการแสดงสัดส่วนของข้อมูล cholesterol Column พบว่า มีจำนวนผู้ป่วยที่มีระดับ cholesterol อยู่ในปกติ(1) มากที่สุด ต่อมามีจำนวนผู้ป่วยที่มีระดับ cholesterol อยู่ในผิดปกติ(2) และจำนวนผู้ป่วยที่มีระดับ cholesterol อยู่ในผิดปกติมาก(3) รองลงมา ตามลำดับ

8. gluc

```
In [104]:
1 gluc_plot = sns.countplot(data = cardio_feature, y = 'gluc',palette="Blues")
2 gluc_plot.set_xlabel("Number of patients")
3 gluc_plot.set_ylabel("Glucose")

Out[104]: Text(0, 0.5, 'Glucose')
```



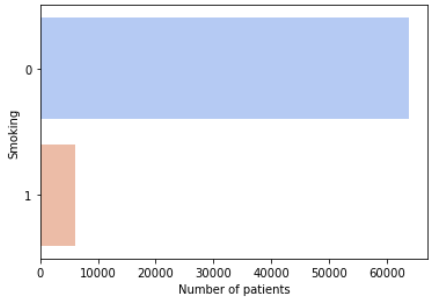
รูปที่ 20 แสดงสัดส่วนของข้อมูล gluc Column

จากการแสดงสัดส่วนของข้อมูล gluc Column พบว่า มีจำนวนผู้ป่วยที่มีระดับน้ำตาลในเลือดอยู่ในปกติ(1) มากที่สุด ต่อมามีจำนวนผู้ป่วยที่มีระดับน้ำตาลในเลือดอยู่ในผิดปกติมาก(3) และจำนวนผู้ป่วยที่มีระดับน้ำตาลในเลือดอยู่ในผิดปกติ(2) รองลงมา ตามลำดับ

9. smoke

```
In [99]:
1 smoke_plot = sns.countplot(data = cardio_feature, y = 'smoke',palette="coolwarm")
2 smoke_plot.set_xlabel("Number of patients")
3 smoke_plot.set_ylabel("Smoking")

Out[99]: Text(0, 0.5, 'Smoking')
```



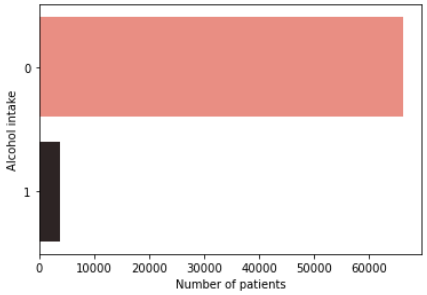
รูปที่ 21 แสดงสัดส่วนของข้อมูล smoke Column

จากการแสดงสัดส่วนของข้อมูล smoke Column พบว่า มีจำนวนผู้ไม่สูบบุหรี่(0) มากกว่าจำนวนผู้ป่วยสูบบุหรี่(1)

10. alco

```
In [115]:
1 alco_plot = sns.countplot(data = cardio_feature, y = 'alco',palette="dark:salmon_r")
2 alco_plot.set_xlabel("Number of patients")
3 alco_plot.set_ylabel("Alcohol intake")

Out[115]: Text(0, 0.5, 'Alcohol intake')
```



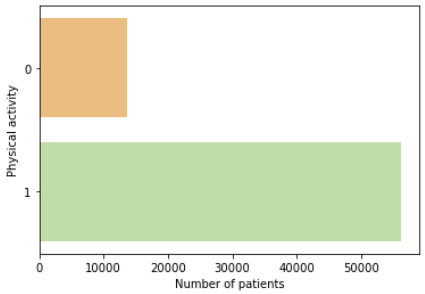
รูปที่ 22 แสดงสัดส่วนของข้อมูล alco Column

จากการแสดงสัดส่วนของข้อมูล alco Column พบว่า มีจำนวนผู้ป่วยไม่ดื่มแอลกอฮอล์(0) มากกว่า จำนวนผู้ป่วยดื่มแอลกอฮอล์(1)

11. active

```
In [118]:
1 active_plot = sns.countplot(data = cardio_feature, y = 'active',palette="Spectral")
2 active_plot.set_xlabel("Number of patients")
3 active_plot.set_ylabel("Physical activity")

Out[118]: Text(0, 0.5, 'Physical activity')
```



รูปที่ 23 แสดงสัดส่วนของข้อมูล active Column

จากการแสดงสัดส่วนของข้อมูล active Column พบว่า มีจำนวนผู้ป่วยออกกำลังกาย(1) มากกว่า จำนวนผู้ป่วยไม่ออกกำลังกาย(0)

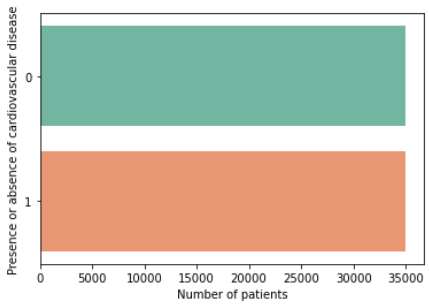
12. cardio

```
In [146]: 1 cardio_df['cardio'].value_counts()

Out[146]: 0    35021
          1    34979
          Name: cardio, dtype: int64

In [147]: 1 cardio_plot = sns.countplot(data = cardio_df, y ='cardio',palette="Set2")
          2 cardio_plot.set_xlabel("Number of patients")
          3 cardio_plot.set_ylabel("Presence or absence of cardiovascular disease")

Out[147]: Text(0, 0.5, 'Presence or absence of cardiovascular disease')
```



รูปที่ 24 แสดงสัดส่วนของข้อมูล cardio Column

จากการแสดงสัดส่วนของข้อมูล cardio Column พบว่า มีจำนวนผู้ป่วยที่ไม่ป่วยเป็นโรคหลอดเลือดหัวใจ(0) และจำนวนผู้ป่วยที่ป่วยเป็นโรคหลอดเลือดหัวใจ(1) ใกล้เคียงกัน

1.2 Data Preparation

- **การจัดการข้อมูลที่มีความผิดปกติใน ap_hi Column หรือ ความดันช่วงหัวใจบีบตัว**

เนื่องจากการทำ Exploratory Data Analysis (EDA) ใน Cardiovascular Disease Dataset พบว่า ap_hi Column มีความผิดปกติ คือ ความดันช่วงหัวใจบีบตัวที่น้อยที่สุดคือ -150 mmHg และมากที่สุดคือ 16,020 mmHg ซึ่งค่าความดันช่วงหัวใจบีบตัวจะติดลบไม่ได้ และไม่ควรน้อยกว่า 80 mmHg และมากกว่า 180 mmHg เพราะจะกลายเป็น emergency case[14]

จึงทำการลบข้อมูลที่มีค่า ความดันช่วงหัวใจบีบตัวต่ำกว่า 80 mmHg และมากกว่า 180 mmHg ออก โดยวิธีดังนี้

```
In [3]: 1 cardio_df.drop(cardio_df[cardio_df['ap_hi'] <= 80].index, inplace=True)

In [5]: 1 cardio_df.drop(cardio_df[cardio_df['ap_hi'] >= 180].index, inplace=True)

In [6]: 1 cardio_df.describe()
```

Out[6]:

	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	a
count	68625.000000	68625.000000	68625.000000	68625.000000	68625.000000	68625.000000	68625.000000	68625.000000	68625.000000	68625.000000	68625.00
mean	19458.395672	1.348736	164.374208	74.085337	126.209996	94.568991	1.362958	1.224233	0.087650	0.053319	0.80
std	2467.999502	0.476574	8.178983	14.280472	15.453699	180.554853	0.677872	0.570224	0.282788	0.224670	0.39
min	10798.000000	1.000000	55.000000	11.000000	85.000000	0.000000	1.000000	1.000000	0.000000	0.000000	0.00
25%	17651.000000	1.000000	159.000000	65.000000	120.000000	80.000000	1.000000	1.000000	0.000000	0.000000	1.00
50%	19697.000000	1.000000	165.000000	72.000000	120.000000	80.000000	1.000000	1.000000	0.000000	0.000000	1.00
75%	21319.000000	2.000000	170.000000	82.000000	140.000000	90.000000	1.000000	1.000000	0.000000	0.000000	1.00
max	23713.000000	2.000000	250.000000	200.000000	179.000000	10000.000000	3.000000	3.000000	1.000000	1.000000	1.00

รูปที่ 25 แสดงขั้นตอนการลบข้อมูลที่มีความผิดปกติ ใน ap_hi Column
และ Descriptive statistics หลังจากลบข้อมูลที่มีความผิดปกติ

- **การจัดการข้อมูลที่มีความผิดปกติใน ap_lo Column หรือ ความดันช่วงหัวใจคลายตัว**

เนื่องจากการทำ Exploratory Data Analysis (EDA) ใน Cardiovascular Disease Dataset พบว่า ap_lo Column มีความผิดปกติ คือ ความดันช่วงหัวใจคลายตัวที่น้อยที่สุดคือ -70 mmHg และมากที่สุดคือ 11,000 ซึ่งค่าความดันช่วงหัวใจคลายตัวจะติดลบไม่ได้ และไม่ควรน้อยกว่า 60 mmHg และมากกว่า 110 mmHg เพราะจะกลายเป็น emergency case[15]

จึงทำการลบข้อมูลที่มีค่า ความดันช่วงหัวใจคลายตัวต่ำกว่า 60 mmHg และมากกว่า 110 mmHg ออก โดยวิธีดังนี้

```
In [7]: 1 cardio_df.drop(cardio_df[cardio_df['ap_lo'] <= 60].index, inplace=True)

In [9]: 1 cardio_df.drop(cardio_df[cardio_df['ap_lo'] >= 110].index, inplace=True)

In [10]: 1 cardio_df.describe()
```

Out[10]:

	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	a
count	64586.000000	64586.000000	64586.000000	64586.000000	64586.000000	64586.000000	64586.000000	64586.000000	64586.000000	64586.000000	64586.00
mean	19492.459217	1.351841	164.455315	74.264562	126.648577	81.790806	1.364460	1.225715	0.087620	0.053185	0.80
std	2458.711906	0.477548	8.149208	14.153589	14.439413	7.736174	0.680408	0.572708	0.282743	0.224404	0.39
min	10859.000000	1.000000	55.000000	11.000000	85.000000	61.000000	1.000000	1.000000	0.000000	0.000000	0.00
25%	17722.250000	1.000000	159.000000	65.000000	120.000000	80.000000	1.000000	1.000000	0.000000	0.000000	1.00
50%	19717.000000	1.000000	165.000000	72.000000	120.000000	80.000000	1.000000	1.000000	0.000000	0.000000	1.00
75%	21343.000000	2.000000	170.000000	82.000000	140.000000	90.000000	1.000000	1.000000	0.000000	0.000000	1.00
max	23713.000000	2.000000	250.000000	200.000000	179.000000	109.000000	3.000000	3.000000	1.000000	1.000000	1.00

รูปที่ 26 แสดงขั้นตอนการลบข้อมูลที่มีความผิดปกติ ใน ap_lo Column และ Descriptive statistics หลังจากลบข้อมูลที่มีความผิดปกติ

- **เพิ่ม BMI Column และจัดการข้อมูล BMI ที่มีความผิดปกติ**

จากการตรวจสอบข้อมูล Weight และ Height พบว่า ส่วนสูงที่น้อยที่สุดคือ 55 cm และมากที่สุดคือ 250 cm น้ำหนักที่น้อยที่สุดคือ 10 kg และมากที่สุดคือ 200 kg จะเห็นว่าช่วงข้อมูลมีค่าผิดปกติจากน้ำหนักและส่วนสูงของมนุษย์ธรรมดาอย่างเห็นได้ชัด[14] ซึ่งเทียบจากค่าดัชนีมวลกาย โดยเอาข้อมูลน้ำหนักและส่วนสูงไปคำนวน

โดยสูตรของ Body Mass Index (BMI) มีดังนี้

$$BMI = \frac{mass_{kg}}{height_m^2},$$

การเพิ่ม BMI Column มีวิธีดังนี้

```
In [46]:
1 cardio_df['BMI'] = cardio_df['weight']/((cardio_df['height']/100)**2)
2 cardio_df
```

Out[46]:

	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio	BMI
0	18393	2	168	62.0	110	80	1	1	0	0	1	0	21.967120
1	20228	1	156	85.0	140	90	3	1	0	0	1	1	34.927679
2	18857	1	165	64.0	130	70	3	1	0	0	0	1	23.507805
3	17623	2	169	82.0	150	100	1	1	0	0	1	1	28.710479
5	21914	1	151	67.0	120	80	2	2	0	0	0	0	29.384676
...	...	...	...	...	...	...	...	...	...	...	...	...	...
69994	21074	1	165	80.0	150	80	1	1	0	0	1	1	29.384757
69995	19240	2	168	76.0	120	80	1	1	1	0	1	0	26.927438
69996	22601	1	158	126.0	140	90	2	2	0	0	1	1	50.472681
69998	22431	1	163	72.0	135	80	1	2	0	0	0	1	27.099251
69999	20540	1	170	72.0	120	80	2	1	0	0	1	0	24.913495

64586 rows × 13 columns

รูปที่ 27 แสดงขั้นตอนการเพิ่ม BMI Column

```
In [47]:
1 cardio_df.describe()
```

Out[47]:

	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio	BMI
count	64586.000000	64586.000000	64586.000000	64586.000000	64586.000000	64586.000000	64586.000000	64586.000000	64586.000000	64586.000000	64586.000000	64586.000000	64586.000000
mean	19492.459217	1.351841	164.455315	74.264562	126.648577	81.790806	1.364460	1.225715	0.087620	0.053185	0.803595	0.498405	27.546129
std	2458.711906	0.477548	8.149208	14.153589	14.439413	7.736174	0.680408	0.572708	0.282743	0.224404	0.397281	0.500001	5.994927
min	10859.000000	1.000000	55.000000	11.000000	85.000000	61.000000	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000	3.471784
25%	17722.250000	1.000000	159.000000	65.000000	120.000000	80.000000	1.000000	1.000000	0.000000	0.000000	1.000000	0.000000	23.875115
50%	19717.000000	1.000000	165.000000	72.000000	120.000000	80.000000	1.000000	1.000000	0.000000	0.000000	1.000000	0.000000	26.397977
75%	21343.000000	2.000000	170.000000	82.000000	140.000000	90.000000	1.000000	1.000000	0.000000	0.000000	1.000000	1.000000	30.119376
max	23713.000000	2.000000	250.000000	200.000000	179.000000	109.000000	3.000000	3.000000	1.000000	1.000000	1.000000	1.000000	298.666667

รูปที่ 28 Descriptive statistics ของ Cardiovascular Disease Dataset หลังจากเพิ่ม BMI Column

จาก Descriptive statistics หลังจากเพิ่ม BMI Column จะเห็นว่า มีความผิดปกติตรงค่า Min และ ค่า Max โดยค่าที่น้อยที่สุด คือ 3.47 และมากที่สุด คือ 298.67 ซึ่งค่า BMI ดังกล่าวผิดปกติไปจากเกณฑ์ของมนุษย์ที่ไม่ควรจะต่ำกว่า 16 และไม่ควรจะสูงกว่า 40[14]

```
In [12]: cardio_df.drop(cardio_df[cardio_df['BMI'] >= 40].index, inplace=True)
In [14]: cardio_df.drop(cardio_df[cardio_df['BMI'] <= 16 ].index, inplace=True)
In [13]: cardio_df.describe()
```

Out[13]:

	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	active	cardio	BMI
count	62925.000000	62925.000000	62925.000000	62925.000000	62925.000000	62925.000000	62925.000000	62925.000000	62925.000000	62925.000000	62925.000000	62925.000000
mean	19485.350687	1.357569	164.658435	73.298499	126.436647	81.694891	1.357441	1.221136	0.088566	0.804243	0.493874	27.049855
std	2460.226549	0.479288	7.786200	12.693376	14.352196	7.698028	0.674749	0.567931	0.284118	0.396785	0.499966	4.435284
min	10859.000000	1.000000	120.000000	11.000000	85.000000	61.000000	1.000000	1.000000	0.000000	0.000000	0.000000	3.471784
25%	17703.000000	1.000000	159.000000	65.000000	120.000000	80.000000	1.000000	1.000000	0.000000	1.000000	0.000000	23.875115
50%	19712.000000	1.000000	165.000000	72.000000	120.000000	80.000000	1.000000	1.000000	0.000000	1.000000	0.000000	26.222685
75%	21337.000000	2.000000	170.000000	81.000000	140.000000	90.000000	1.000000	1.000000	0.000000	1.000000	1.000000	29.757785
max	23713.000000	2.000000	250.000000	135.000000	179.000000	109.000000	3.000000	3.000000	1.000000	1.000000	1.000000	39.965649

รูปที่ 29 แสดงขั้นตอนการลบข้อมูลที่มีความผิดปกติ ใน BMI
และ Descriptive statistics หลังจากลบข้อมูลที่มีความผิดปกติ

- **การเปลี่ยนแปลงข้อมูลใน gender, cholesterol, gluc Column**

เปลี่ยนแปลงข้อมูลในทั้ง 3 Column ดังนี้ gender, cholesterol และ gluc เพื่อให้เข้าใจง่ายขึ้น และสะดวกต่อการนำไปวิเคราะห์ต่อ

```
In [32]: cardio_df['gender'] = cardio_df['gender'].replace({1:0, 2:1})
         cardio_df['cholesterol'] = cardio_df['cholesterol'].replace({1:0, 2:1, 3:2})
         cardio_df['gluc'] = cardio_df['gluc'].replace({1:0, 2:1, 3:2})
In [33]: cardio_df
```

Out[33]:

	age	gender	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio	BMI
0	18393	1	110	80	0	0	0	0	1	0	21.967120
1	20228	0	140	90	2	0	0	0	1	1	34.927679
2	18857	0	130	70	2	0	0	0	0	1	23.507805
3	17623	1	150	100	0	0	0	0	1	1	28.710479
4	21914	0	120	80	1	1	0	0	0	0	29.384676
...	...	...	...	...	...	...	...	...	...	...	...
62874	19699	0	130	90	0	0	0	0	1	1	23.661439
62875	21074	0	150	80	0	0	0	0	1	1	29.384757
62876	19240	1	120	80	0	0	1	0	1	0	26.927438
62877	22431	0	135	80	0	1	0	0	0	1	27.099251
62878	20540	0	120	80	1	0	0	0	1	0	24.913495

62879 rows × 11 columns

รูปที่ 30 แสดงขั้นตอนการเปลี่ยนแปลงข้อมูลในทั้ง 3 Column

- **การเปลี่ยนแปลงหน่วยของข้อมูลใน age Column**

เปลี่ยนแปลงหน่วยของข้อมูลใน age Column จากหน่วยวัน ให้เป็น ปี เพื่อให้เข้าใจง่ายขึ้น และสะดวกต่อการนำไปวิเคราะห์ต่อ

In [34]:
```
cardio_df["age"] = cardio_df["age"].apply(lambda x : x/365)
cardio_df
```

Out[34]:

	age	gender	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio	BMI
0	50.391781	1	110	80	0	0	0	0	1	0	21.967120
1	55.419178	0	140	90	2	0	0	0	1	1	34.927679
2	51.663014	0	130	70	2	0	0	0	0	1	23.507805
3	48.282192	1	150	100	0	0	0	0	1	1	28.710479
4	60.038356	0	120	80	1	1	0	0	0	0	29.384676
...	...	...	...	...	...	...	...	...	...	...	...
62874	53.969863	0	130	90	0	0	0	0	1	1	23.661439
62875	57.736986	0	150	80	0	0	0	0	1	1	29.384757
62876	52.712329	1	120	80	0	0	1	0	1	0	26.927438
62877	61.454795	0	135	80	0	1	0	0	0	1	27.099251
62878	56.273973	0	120	80	1	0	0	0	1	0	24.913495

62879 rows × 11 columns

รูปที่ 31 แสดงขั้นตอนการเปลี่ยนแปลง age Column ที่มีหน่วยเป็นวัน ให้เป็น ปี

1.3 สรุปข้อมูลหลังจากทำ Data Preparation

In [1]:
```
import pandas as pd
cardio_train_cleaned = pd.read_csv('cardio_train_cleaned_2.csv')
cardio_train_cleaned.head()
```

Out[1]:

	age	gender	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio	BMI
0	50.391781	1	110	80	0	0	0	0	1	0	21.967120
1	55.419178	0	140	90	2	0	0	0	1	1	34.927679
2	51.663014	0	130	70	2	0	0	0	0	1	23.507805
3	48.282192	1	150	100	0	0	0	0	1	1	28.710479
4	60.038356	0	120	80	1	1	0	0	0	0	29.384676

In [2]:
```
cardio_train_cleaned.describe()
```

Out[2]:

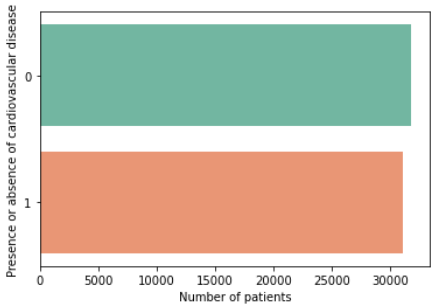
	age	gender	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio	BMI
count	62879.000000	62879.000000	62879.000000	62879.000000	62879.000000	62879.000000	62879.000000	62879.000000	62879.000000	62879.000000	62879.000000
mean	53.384165	0.357607	126.441562	81.697562	0.357433	0.221107	0.088583	0.053261	0.804227	0.493933	27.059374
std	6.740391	0.479300	14.349783	7.695989	0.674775	0.567930	0.284143	0.224555	0.396798	0.499967	4.422347
min	29.750685	0.000000	85.000000	61.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	16.003658
25%	48.501370	0.000000	120.000000	80.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	23.875115
50%	54.005479	0.000000	120.000000	80.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	26.233556
75%	58.457534	1.000000	140.000000	90.000000	0.000000	0.000000	0.000000	0.000000	1.000000	1.000000	29.757785
max	64.967123	1.000000	179.000000	109.000000	2.000000	2.000000	1.000000	1.000000	1.000000	1.000000	39.965649

In [5]:
```
cardio_train_cleaned['cardio'].value_counts()
```

Out[5]:
```
0    31821
1    31058
Name: cardio, dtype: int64
```

In [6]:
```
import seaborn as sns
cardio_plot = sns.countplot(data = cardio_train_cleaned, y ='cardio',palette="Set2")
cardio_plot.set_xlabel("Number of patients")
cardio_plot.set_ylabel("Presence or absence of cardiovascular disease")
```

Out[6]:
```
Text(0, 0.5, 'Presence or absence of cardiovascular disease')
```



รูปที่ 32 แสดงสรุปข้อมูลหลังจากทำ Data Preparation และ การกระจายของ cardio Column

จากการกระจายของ cardio Column หลังจากการทำ Data Preparation แล้ว พบว่า สัดส่วนของข้อมูลใน cardio Column มีค่าใกล้เคียงกัน ดังนั้นจึงสามารถใช้ ค่าความถูกต้อง (Accuracy) ในการประเมินแบบจำลองในขั้นตอนต่อ ๆ ไปได้

2. Synthetic Dataset

```
from sklearn.datasets import make_classification
x_synthetic, y_synthetic = make_classification(n_samples= 60000, n_features=10, shuffle=True, random_state=42)
```

```
x_synthetic
```

```
array([[-2.11899452e+00, -8.48017636e-01,  6.17129065e-01, ...,
        -2.77552417e+00, -1.50515129e+00, -8.59804705e-02],
       [-7.29878051e-01,  4.46479495e-01, -7.45519169e-02, ...,
         4.86786675e-01, -5.12082710e-01,  5.00598078e-01],
       [ 1.45371877e-01, -2.60005959e-01,  1.93477527e+00, ...,
        -8.36997210e-01,  6.69408764e-01, -1.72959947e+00],
       ...,
       [-3.27944022e-01,  1.05149551e-01, -1.19621579e+00, ...,
         2.95799523e-02, -1.40240674e-03,  2.63721824e-01],
       [ 3.67870218e-01,  1.93080714e-01, -1.09923089e+00, ...,
        -5.67475090e-01,  1.26419801e+00,  3.30461192e-01],
       [-1.69931506e+00,  4.11706992e-01,  1.52889908e+00, ...,
        -4.19587340e-01, -4.90841759e-01, -7.34520553e-01]])
```

```
y_synthetic
```

```
array([1, 0, 1, ..., 1, 1, 0])
```

```
# split train and test set
x_synthetic_train, x_synthetic_test, y_synthetic_train, y_synthetic_test = train_test_split(x_synthetic,
                                                                                            y_synthetic, test_size = 0.3)
```

รูปที่ 33 แสดงข้อมูลที่สร้างขึ้นจากไลบรารี่ make_classification

บทที่ 5
ผลลัพธ์การวิเคราะห์

1. ผลลัพธ์การวิเคราะห์จาก Cardiovascular Disease Dataset

1.1 Logistic Regression Classifier โดยการใช้ GridSearchCV()

```
from sklearn.linear_model import LogisticRegression

Logres_classifier = LogisticRegression()
Logres_classifier.get_params()
```

```
{'C': 1.0,
 'class_weight': None,
 'dual': False,
 'fit_intercept': True,
 'intercept_scaling': 1,
 'l1_ratio': None,
 'max_iter': 100,
 'multi_class': 'auto',
 'n_jobs': None,
 'penalty': 'l2',
 'random_state': None,
 'solver': 'lbfgs',
 'tol': 0.0001,
 'verbose': 0,
 'warm_start': False}
```

รูปที่ 34 แสดงโมเดลและพารามิเตอร์ของ Logistic Regression

```
from sklearn.model_selection import GridSearchCV

params = dict()
params['C'] = [0.001, 0.01, 0.1, 1, 10, 100, 1000]
params["solver"] = ['liblinear', 'newton-cg', 'lbfgs', 'sag', 'saga']
params['max_iter'] = [2500, 5000, 7500, 10000]

gridsearch = GridSearchCV(Logres_classifier, params, scoring='accuracy', verbose=3)
gridsearch.fit(x_train, y_train)
```

```
Fitting 5 folds for each of 140 candidates, totalling 700 fits

GridSearchCV(estimator=LogisticRegression(),
             param_grid={'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000],
                         'max_iter': [2500, 5000, 7500, 10000],
                         'solver': ['liblinear', 'newton-cg', 'lbfgs', 'sag',
                                    'saga']},
             scoring='accuracy', verbose=3)
```

รูปที่ 35 แสดงการหาพารามิเตอร์ที่ดีที่สุดสำหรับ Logistic Regression ด้วย GridSearchCV()

การฝึกโมเดล Logistic Regression โดยนำอัลกอริทึม GridSearchCV() มาใช้ในการหาพารามิเตอร์ที่เหมาะสม ที่ทำให้โมเดลได้ค่าความแม่นยำสูงสุด ซึ่งพารามิเตอร์ที่ทำการปรับค่า มีดังนี้

1. **C :** Inverse of regularization strength
   Regularization คือ เทคนิคในการปรับโมเดลเพื่อลดค่า loss function และป้องกันไม่ให้โมเดลจำเพาะต่อข้อมูลน้อยหรือมากเกินไป (underfitting or overfitting)
   ดังนั้นเมื่อเรากำหนด ค่า C ให้มีค่าสูง เป็นการให้ความสำคัญ (weight) กับชุดข้อมูลที่ใช้ฝึกโมเดล [16][17]
2. **solver :** อัลกอริทึมที่ใช้ในขั้นตอนการลดค่า loss function หรือกระบวนการ optimization
3. **max_iter :** จำนวนรอบสูงสุดในการดำเนินการตาม solver เพื่อหาจุดที่ค่า loss function ไม่เปลี่ยนแปลงค่า (converge)[18]

```
print('Best Hyperparameters: %s' % gridsearch_.best_params_)
```

```
Best Hyperparameters: {'C': 0.1, 'max_iter': 2500, 'solver': 'liblinear'}
```

รูปที่ 36 แสดงการพารามิเตอร์ที่ดีที่สุดสำหรับ Logistic Regression ด้วย GridSearchCV()

การฝึกโมเดล Logistic Regression โดยนำอัลกอริทึม GridSearchCV() ใช้ทั้งหมด 140 combinations โดยฝึกโมเดลด้วยชุดข้อมูลที่แบ่งด้วยเทคนิค Cross Validation ออกเป็น 5 ส่วน ผลลัพธ์ combination ที่ดีที่สุดคือ `{'C': 0.1, 'max_iter': 2500, 'solver': 'liblinear'}`

```
lr = gridsearch.best_estimator_
```

```
acc = round(lr.score(x_train, y_train) * 100, 2)
acc_test = round(lr.score(x_test, y_test) * 100, 2)
print(acc, acc_test)
```

```
72.17 72.66
```

รูปที่ 37 แสดงผลลัพธ์ค่าความแม่นยำของ Logistic Regression

Logistic regression combination ที่ดีที่สุดได้ค่าความแม่นยำของชุดข้อมูลฝึกและชุดข้อมูลทดสอบเท่ากับ 72.17% และ 72.66% ตามลำดับ

```
lr_result = lr.predict(x_test)
print(classification_report(y_true=y_test, y_pred=lr_result))
```

```
              precision    recall  f1-score   support

           0       0.71      0.79      0.74      9573
           1       0.75      0.66      0.71      9291

    accuracy                           0.73     18864
   macro avg       0.73      0.73      0.73     18864
weighted avg       0.73      0.73      0.73     18864
```

รูปที่ 38 แสดง confusion matrix ของโมเดล Logistic Regression

1.2 Random Forest Classifier โดยการใช้ GridSearchCV()

```
%%time
from sklearn.ensemble import RandomForestClassifier
Randfor_classifier = RandomForestClassifier()
Randfor_classifier.get_params()
```

```
Wall time: 0 ns

{'bootstrap': True,
 'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': None,
 'max_features': 'auto',
 'max_leaf_nodes': None,
 'max_samples': None,
 'min_impurity_decrease': 0.0,
 'min_impurity_split': None,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'n_estimators': 100,
 'n_jobs': None,
 'oob_score': False,
 'random_state': None,
 'verbose': 0,
 'warm_start': False}
```

รูปที่ 39 แสดงโมเดลและพารามิเตอร์ของ Random Forest

```
params = dict()
params['n_estimators'] = [100, 200, 300]
params['max_depth'] = [2, 4, 8, 16, None]
params["max_features"] = ['auto', 'sqrt']
params["min_samples_leaf"] = [1, 2, 3]
params["min_samples_split"] = [2, 5, 10]
params["criterion"] = ["gini", "entropy"]
params['class_weight'] = ['balanced','balanced_subsample']
params['bootstrap'] = [True]
```

```
gridsearch = GridSearchCV(Randfor_classifier, params, scoring='accuracy', verbose=3)
gridsearch.fit(x_train, y_train)
```

```
Fitting 5 folds for each of 1080 candidates, totalling 5400 fits

GridSearchCV(estimator=RandomForestClassifier(),
             param_grid={'bootstrap': [True],
                         'class_weight': ['balanced', 'balanced_subsample'],
                         'criterion': ['gini', 'entropy'],
                         'max_depth': [2, 4, 8, 16, None],
                         'max_features': ['auto', 'sqrt'],
                         'min_samples_leaf': [1, 2, 3],
                         'min_samples_split': [2, 5, 10],
                         'n_estimators': [100, 200, 300]},
             scoring='accuracy', verbose=3)
```

รูปที่ 40 แสดงการหาพารามิเตอร์ที่ดีที่สุดสำหรับ Random Forest ด้วย GridSearchCV()

การฝึกโมเดล Random Forest โดยนำอัลกอริทึม GridSearchCV() มาใช้ในการหาพารามิเตอร์ที่เหมาะสม ที่ทำให้โมเดลได้ค่าความแม่นยำสูงสุด ซึ่งพารามิเตอร์ที่ทำการปรับค่า มีดังนี้

1. **n_estimators :** จำนวนของต้นไม้ที่ใช้ในการทำนาย
2. **max_depth :** ความลึกสูงสุดของต้นไม้ การเพิ่มค่านี้จะทำให้โมเดลซับซ้อนขึ้น
3. **max_features :** จำนวน feature ที่ใช้พิจารณาในการแบ่ง node
4. **min_samples_leaf :** จำนวนข้อมูลที่น้อยที่สุดใน 1 leaf node
5. **min_samples_split :** จำนวนข้อมูลที่น้อยที่สุดในการแบ่ง node
6. **criterion :** ฟังก์ชั่นที่ใช้ในการวัดความสามารถในการแบ่ง
7. **class_weight :** ค่า weight สำหรับแต่ละ class
8. **bootstrap :** เทคนิคในการใช้ bootstrap samples ในการสร้างต้นไม้[19][20]

```
print('Best Hyperparameters: %s' % gridsearch.best_params_)
```

```
Best Hyperparameters: {'bootstrap': True, 'class_weight': 'balanced', 'criterion': 'gini', 'max_depth': 8, 'max_features': 'sqr
t', 'min_samples_leaf': 3, 'min_samples_split': 2, 'n_estimators': 200}
```

รูปที่ 41 แสดงการพารามิเตอร์ที่ดีที่สุดสำหรับ Logistic Regression ด้วย GridSearchCV()

การฝึกโมเดล Random Forest โดยนำอัลกอริทึม GridSearchCV() ใช้ทั้งหมด 1080 combinations โดยฝึกโมเดลด้วยชุดข้อมูลที่แบ่งด้วยเทคนิค Cross Validation ออกเป็น 5 ส่วน ผลลัพธ์ combination ที่ดีที่สุดคือ {'bootstrap': True, 'class_weight': 'balanced', 'criterion': 'gini', 'max_depth': 8, 'max_features': 'sqrt', 'min_samples_leaf': 3, 'min_samples_split': 2, 'n_estimators': 200}

```
rf = gridsearch.best_estimator_

acc = round(rf.score(x_train, y_train) * 100, 2)
acc_test = round(rf.score(x_test, y_test) * 100, 2)
print(acc, acc_test)
```

```
73.75 73.26
```

รูปที่ 42 แสดงผลลัพธ์ค่าความแม่นยำของ Logistic Regression

Random Forest combination ที่ดีที่สุดได้ค่าความแม่นยำของชุดข้อมูลฝึกและชุดข้อมูลทดสอบเท่ากับ 73.75% และ 73.26% ตามลำดับ

```
rf_result = rf.predict(x_test)
print(classification_report(y_true=y_test, y_pred=rf_result))
```

```
              precision    recall  f1-score   support

           0       0.72      0.79      0.75      9573
           1       0.75      0.68      0.71      9291

    accuracy                           0.73     18864
   macro avg       0.73      0.73      0.73     18864
weighted avg       0.73      0.73      0.73     18864
```

รูปที่ 43 แสดง confusion matrix ของโมเดล Random Forest

1.3 eXtreme Gradient Boosting (xGB) โดยการใช้ GridSearchCV()

```python
from sklearn.model_selection import GridSearchCV
import xgboost

colsample_bytree = [0.3, 0.5, 1.0]
gamma = [0.1, 1, 1.5]
learning_rate = [0.001, 0.01]
min_child_weight = [1, 5, 10]
scale_pos_weight = [1, 2, 4]
n_estimators = [50, 100]
max_depth = [5, 10]

param_grid = dict(
    colsample_bytree=colsample_bytree,
    gamma=gamma,
    learning_rate=learning_rate,
    min_child_weight=min_child_weight,
    scale_pos_weight=scale_pos_weight,
    n_estimators=n_estimators,
    max_depth=max_depth )

model = XGBClassifier()

grid_search = GridSearchCV(model,
                           param_grid=param_grid,
                           scoring='accuracy',
                           verbose=3, cv = 3 )

best_model = grid_search.fit(x_train, y_train)
print('Optimum parameters', best_model.best_params_)
```

```
Fitting 3 folds for each of 648 candidates, totalling 1944 fits
```

```
Optimum parameters {'colsample_bytree': 0.5, 'gamma': 1, 'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight': 1, 'n_estim
ators': 100, 'scale_pos_weight': 1}
```

รูปที่ 44 แสดง eXtreme Gradient Boosting (xGB) โดยการใช้ GridSearchCV()

```
In [20]: print('Optimum parameters', best_model.best_params_)

Optimum parameters {'colsample_bytree': 0.5, 'gamma': 1, 'learning_rate': 0.01, 'max_depth': 5, 'min_child_weight': 1, 'n_estimators': 100, 'scale_pos_weight': 1}
```

```
In [26]: xgboost.get_params()

Out[26]: {'objective': 'binary:logistic',
 'use_label_encoder': False,
 'base_score': 0.5,
 'booster': 'gbtree',
 'callbacks': None,
 'colsample_bylevel': 1,
 'colsample_bynode': 1,
 'colsample_bytree': 0.5,
 'early_stopping_rounds': None,
 'enable_categorical': False,
 'eval_metric': None,
 'gamma': 1,
 'gpu_id': -1,
 'grow_policy': 'depthwise',
 'importance_type': None,
 'interaction_constraints': '',
 'learning_rate': 0.01,
 'max_bin': 256,
 'max_cat_to_onehot': 4,
 'max_delta_step': 0,
 'max_depth': 5,
 'max_leaves': 0,
 'min_child_weight': 1,
 'missing': nan,
 'monotone_constraints': '()',
 'n_estimators': 100,
 'n_jobs': 0,
 'num_parallel_tree': 1,
 'predictor': 'auto',
 'random_state': 0,
 'reg_alpha': 0,
 'reg_lambda': 1,
 'sampling_method': 'uniform',
 'scale_pos_weight': 1,

 'subsample': 1,
 'tree_method': 'exact',
 'validate_parameters': 1,
 'verbosity': None}
```

รูปที่ 45 แสดง Best Parameter จากการใช้ GridSearchCV()

การฝึกโมเดล eXtreme Gradient Boosting (xGB) โดยนำอัลกอริทึม gridsearchcv มาใช้ในการหาพารามิเตอร์ที่เหมาะสม ที่ทำให้โมเดลได้ค่าความแม่นยำสูงสุด ซึ่งพารามิเตอร์ที่ทำการปรับค่า มีดังนี้

1.colsample_bytree : คืออัตราส่วน subsample ของคอลัมน์เมื่อสร้างต้นไม้แต่ละต้น การสุ่มตัวอย่างจะเกิดขึ้นครั้งเดียวสำหรับต้นไม้ทุกต้นที่สร้าง

2.gamma : คือการลด Minimum loss ในการแบ่งเพิ่มเติมบน leaf node ของต้นไม้ ยิ่งแกมมามากเท่าไร อัลกอริธึมก็จะยิ่งอนุรักษ์ค่า gamma มากขึ้นเท่านั้น

3.learning_rate : เป็น parameter ที่ควบคุมว่าจะเปลี่ยนแปลงค่า weight มากหรือน้อยเท่าไหร่ของ Model ใน 1 Step ของ Training ถ้าปรับไว้มากเกินไปจะทำให้คำตอบของ Model ไม่ลู่เข้าสู่คำตอบจริงหรือจุดที่เป็น Global Minimum แต่ถ้าปรับไว้น้อยเกินไปจะทำให้ใช้เวลาในการ Train นานขึ้น

4.Max_depth : ความลึกสูงสุดของต้นไม้ การเพิ่มค่านี้จะทำให้โมเดลซับซ้อนขึ้น ถ้ากำหนดค่า Max_depth = 0 หมายถึงไม่จำกัดความลึก ซึ่งสิ่งที่ควรต้องระวังคือ XGBoost จะใช้หน่วยความจำเยอะ ดังนั้นจึงไม่ควรกำหนดให้ Max_depth = 0

5.min_child_weight : ผลรวมขั้นต่ำของน้ำหนักอินสแตนซ์ที่จำเป็นสำหรับโหนดลูก หากการแบ่งต้นไม้ให้ค่าน้ำหนักรวมของอินสแตนซ์โหนดปลายสุดน้อยกว่า min_child_weight จะยกเลิกการแบ่งเพิ่มเติม

6.n_estimators : จำนวนของต้นไม้ที่ใช้ในการทำนาย

7.scale_pos_weight : ควบคุมความสมดุลของน้ำหนักบวกและลบ ซึ่งมีประโยชน์สำหรับ class ที่ไม่สมดุล[21]

```
xgboost = best_model.best_estimator_
xgboost.fit(x_train, y_train)

acc_train = round(xgboost.score(x_train,y_train) * 100, 4)
acc_test = round(xgboost.score(x_test, y_test)* 100, 4)
print(acc_train, acc_test)
```

```
73.2159 72.7258
```

```
from sklearn.metrics import classification_report

result = xgboost.predict(x_test)
print(classification_report(y_true=y_test, y_pred=result))
```

```
              precision    recall  f1-score   support

           0       0.70      0.79      0.74      9416
           1       0.76      0.66      0.71      9448

    accuracy                           0.73     18864
   macro avg       0.73      0.73      0.73     18864
weighted avg       0.73      0.73      0.73     18864
```

รูปที่ 46 แสดงค่าความถูกต้อง (Accuracy) ของ train และ test data

จากการทำ eXtreme Gradient Boosting โดยการใช้ GridSearchCV() เพื่อหา parameter ที่ดีที่สุดจากการ Tuning Parameters วนซ้ำไปเรื่อย ๆ จนได้ Best parameter คือ { 'colsample_bytree': 0.5, 'gamma': 1.5, 'learning_rate': 0.01, 'max_depth': 10, 'min_child_weight': 10, 'n_estimators': 100, 'scale_pos_weight': 1 } ได้ค่าความแม่นยำของชุดข้อมูลฝึกและชุดข้อมูลทดสอบเท่ากับ 74.4292% และ 72.8% ตามลำดับ

2. ผลลัพธ์การวิเคราะห์จาก Synthetic Dataset

จากการใช้อัลกอริทึมทั้ง 3 ได้แก่ Logistic Regression, Random Forest และ eXtreme Gradient Boosting สร้างแบบจำลองจากข้อมูล Synthetic Dataset โดยการกำหนดค่า hyperparameters แบบเดียวกับผลลัพธ์จากการทำ GridSearchCV() สรุปผลลัพธ์มีค่าดังต่อไปนี้

1. Logistic Regression

```
lr_syn = LogisticRegression(C=0.1, max_iter=2500, solver='liblinear')
lr_syn.fit(x_synthetic_train, y_synthetic_train)
```

```
LogisticRegression(C=0.1, max_iter=2500, solver='liblinear')
```

```
acc = round(lr_syn.score(x_synthetic_train, y_synthetic_train) * 100, 2)
acc_test = round(lr_syn.score(x_synthetic_test, y_synthetic_test) * 100, 2)
print(acc, acc_test)
```

```
88.93 89.2
```

รูปที่ 47 แสดงค่าผลลัพธ์ของโมเดล Logistic Regrssion กับชุดข้อมูลสังเคราะห์

Logistic regression combination ที่ดีที่สุดได้ค่าความแม่นยำของชุดข้อมูลฝึกและชุดข้อมูลทดสอบเท่ากับ 88.93% และ 89.2% ตามลำดับ

2. Random Forest

```
rf_syn = RandomForestClassifier(bootstrap=True, class_weight='balanced',
                                criterion='gini', max_depth=8,
                                max_features='sqrt', min_samples_leaf=3,
                                min_samples_split=2, n_estimators=200)
rf_syn.fit(x_synthetic_train, y_synthetic_train)
```

```
RandomForestClassifier(class_weight='balanced', max_depth=8,
                       max_features='sqrt', min_samples_leaf=3,
                       n_estimators=200)
```

```
acc = round(rf_syn.score(x_synthetic_train, y_synthetic_train) * 100, 2)
acc_test = round(rf_syn.score(x_synthetic_test, y_synthetic_test) * 100, 2)
print(acc, acc_test)
```

```
92.01 91.61
```

รูปที่ 48 แสดงค่าผลลัพธ์ของโมเดล Random Forest กับชุดข้อมูลสังเคราะห์

Random Forest combination ที่ดีที่สุดได้ค่าความแม่นยำของชุดข้อมูลฝึกและชุดข้อมูลทดสอบเท่ากับ 88.93% และ 89.2% ตามลำดับ

3. eXtreme Gradient Boosting

```
xgb_syn = xgboost.XGBClassifier(colsample_bytree = 0.5, gamma = 1,
                                learning_rate = 0.01, min_child_weight = 1,
                                max_depth = 5, scale_pos_weight = 1, n_estimators=100)
xgb_syn.fit(x_synthetic_train, y_synthetic_train)
```

```
XGBClassifier(base_score=0.5, booster='gbtree', callbacks=None,
              colsample_bylevel=1, colsample_bynode=1, colsample_bytree=0.5,
              early_stopping_rounds=None, enable_categorical=False,
              eval_metric=None, gamma=1, gpu_id=-1, grow_policy='depthwise',
              importance_type=None, interaction_constraints='',
              learning_rate=0.01, max_bin=256, max_cat_to_onehot=4,
              max_delta_step=0, max_depth=5, max_leaves=0, min_child_weight=1,
              missing=nan, monotone_constraints='()', n_estimators=100,
              n_jobs=0, num_parallel_tree=1, predictor='auto', random_state=0,
              reg_alpha=0, reg_lambda=1, ...)
```

```
acc = round(xgb_syn.score(x_synthetic_train, y_synthetic_train) * 100, 2)
acc_test = round(xgb_syn.score(x_synthetic_test, y_synthetic_test) * 100, 2)
print(acc, acc_test)
```

```
91.41 91.41
```

รูปที่ 49 แสดงค่าผลลัพธ์ของโมเดล eXtreme Gradient Boosting กับชุดข้อมูล

eXtreme Gradient Boosting combination ที่ดีที่สุดได้ค่าความแม่นยำของชุดข้อมูลฝึกและชุดข้อมูลทดสอบเท่ากับ 91.41% และ 91.41% ตามลำดับ

บทที่ 6
สรุปและการอภิปรายผลการวิเคราะห์

การวิเคราะห์ผลจาก Cardiovascular Disease Dataset จะได้ว่า

อัลกอริทึมทั้ง 3 คือ Logistic Regression โดยการใช้ GridSearchCV() , Random Forest โดยการใช้ GridSearchCV() และ eXtreme Gradient Boosting โดยการใช้ GridSearchCV() สร้างแบบจำลองเพื่อใช้ในการทำนายความเสี่ยงการเป็นโรคหลอดเลือดหัวใจจากชุดข้อมูล Cardiovascular Disease สรุปผลได้ดังต่อไปนี้

สำหรับค่าความถูกต้อง (Accuracy) ชุดข้อมูลฝึกและชุดข้อมูลทดสอบของแบบจำลอง Logistic Regression โดยการใช้ GridSearchCV() เท่ากับ 72.17% และ 72.66% ตามลำดับ

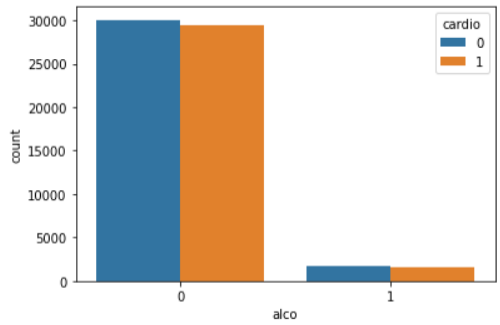
สำหรับค่าความถูกต้อง (Accuracy) ชุดข้อมูลฝึกและชุดข้อมูลทดสอบของแบบจำลอง Random Forest โดยการใช้ GridSearchCV() เท่ากับ 73.75% และ 73.26% ตามลำดับ

สำหรับค่าความถูกต้อง (Accuracy) ชุดข้อมูลฝึกและชุดข้อมูลทดสอบของแบบจำลอง eXtreme Gradient Boosting โดยการใช้ GridSearchCV() เท่ากับ 74.4292% และ 72.8% ตามลำดับ

ซึ่งค่าความถูกต้องที่ได้จากแบบจำลองแต่ละอัลกอริทึมพบว่าค่าค่อนข้างใกล้เคียงกัน จากการสำรวจข้อมูลหรือทำ Exploratory Data Analysis (EDA) จึงพบว่า มีเพียงบางปัจจัยในชุดข้อมูลที่ส่งผลต่อการแยก class ได้อย่างชัดเจน สาเหตุต่อมาอาจมาจากค่าจากปัจจัยในชุดข้อมูลที่เก็บมาผิดประเภท เช่น ค่าโคเลสเตอรอลและระดับน้ำตาลในเลือด ที่ควรจะเก็บมาเป็นค่า numerical แต่ข้อมูลชุดนี้กลับเก็บค่าเป็น categorical แบบมีลำดับ สาเหตุสุดท้ายคือข้อมูลชุดนี้ไม่มีแหล่งที่มาอ้างอิง ซึ่งส่งผลต่อความน่าเชื่อถือของชุดข้อมูล จากทั้ง 3 สาเหตุดังกล่าวอาจเป็นที่มาที่ส่งผลต่อประสิทธิภาพในการแยกประเภทของแบบจำลองหรือค่าความถูกต้อง

```
1 sns.countplot(x='alco', hue='cardio', data=df)
```

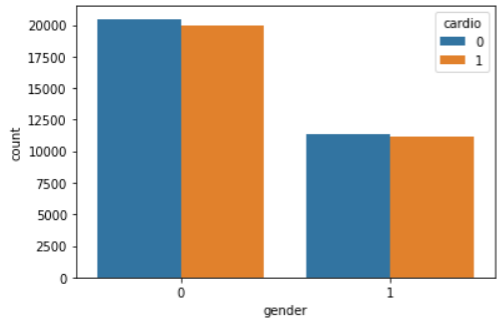
```
<AxesSubplot:xlabel='alco', ylabel='count'>
```



รูปที่ 50 แสดงสัดส่วนข้อมูลการดื่มแอลกอฮอล์ระหว่างผู้ที่ไม่ป่วยและผู้ที่ป่วยเป็นโรคหลอดเลือดหัวใจ

```
1 sns.countplot(x='gender', hue='cardio', data=df)
```

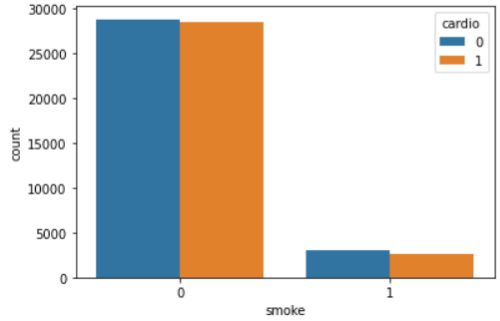
```
<AxesSubplot:xlabel='gender', ylabel='count'>
```



รูปที่ 51 แสดงสัดส่วนข้อมูลเพศระหว่างผู้ที่ไม่ป่วยและผู้ที่ป่วยเป็นโรคหลอดเลือดหัวใจ

```
1 sns.countplot(x='smoke', hue='cardio', data=df)
```

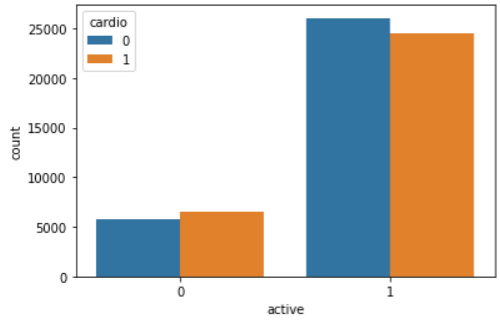
```
<AxesSubplot:xlabel='smoke', ylabel='count'>
```



รูปที่ 52 แสดงสัดส่วนข้อมูลการสูบบุหรี่ระหว่างผู้ที่ไม่ป่วยและผู้ที่ป่วยเป็นโรคหลอดเลือดหัวใจ

```
1 sns.countplot(x='active', hue='cardio', data=df)
```

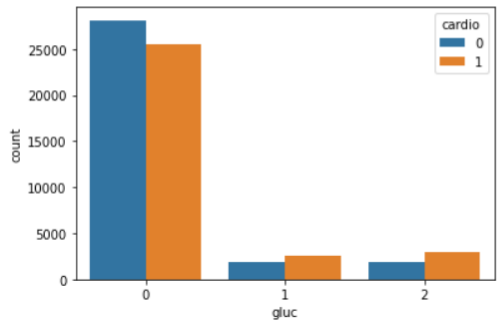
```
<AxesSubplot:xlabel='active', ylabel='count'>
```



รูปที่ 53 แสดงสัดส่วนข้อมูลการออกกำลังระหว่างผู้ที่ไม่ป่วยและผู้ที่ป่วยเป็นโรคหลอดเลือดหัวใจ

```
1 sns.countplot(x='gluc', hue='cardio', data=df)
```

```
<AxesSubplot:xlabel='gluc', ylabel='count'>
```



รูปที่ 54 แสดงสัดส่วนข้อมูลระดับน้ำตาลในเลือดระหว่างผู้ที่ไม่ป่วยและผู้ที่ป่วยเป็นโรคหลอดเลือดหัวใจ

การวิเคราะห์ผลจาก Synthetic Dataset จะได้ว่า

อัลกอริทึมทั้ง 3 ได้แก่ Logistic Regression, Random Forest และ eXtreme Gradient Boosting สร้างแบบจำลองจากข้อมูล Synthetic Dataset โดยการกำหนดค่า hyperparameters แบบเดียวกับผลลัพธ์จากการทำ GridSearchCV() สรุปผลลัพธ์มีค่าดังต่อไปนี้

สำหรับค่าความถูกต้อง (Accuracy) ชุดข้อมูลฝึกและชุดข้อมูลทดสอบของแบบจำลอง Logistic Regression เท่ากับ 88.93% และ 89.20% ตามลำดับ

สำหรับค่าความถูกต้อง (Accuracy) ชุดข้อมูลฝึกและชุดข้อมูลทดสอบของแบบจำลอง Random Forest เท่ากับ 92.01% และ 91.61% ตามลำดับ

สำหรับค่าความถูกต้อง (Accuracy) ชุดข้อมูลฝึกและชุดข้อมูลทดสอบของแบบจำลอง eXtreme Gradient Boosting เท่ากับ 91.41% และ 91.41% ตามลำดับ

Dataset	No.	Algorithms	Accuracy of Train set	Accuracy of Test set
Cardiovascular Disease Dataset	1	Logistic Regression Classifier โดยการใช้ GridSearchCV()	72.17%	72.66%
	2	Random Forest Classifier โดยการใช้ GridSearchCV()	73.75%	73.26%
	3	eXtreme Gradient Boosting โดยการใช้ GridSearchCV()	74.4292%	72.8%
Synthetic Dataset	1	Logistic Regression Classifier โดยการใช้ GridSearchCV()	88.93%	89.20%
	2	Random Forest Classifier โดยการใช้ GridSearchCV()	92.01%	91.61%
	3	eXtreme Gradient Boosting โดยการใช้ GridSearchCV()	91.41%	91.41%

ตารางที่ 1 แสดงค่าความถูกต้อง (Accuracy) ของทั้ง 3 อัลกอริทึม ในทั้ง 2 Dataset

บรรณานุกรม

1. โรคหัวใจและหลอดเลือด [อินเตอร์เน็ต]. [เข้าถึงเมื่อ 14 พ.ค. 2565]. เข้าถึงได้จาก: https://ccpe.pharmacycouncil.org/index.php?option=seminar_detail&subpage=seminar_detail&id=1623
2. รอบรู้เรื่องโรคและภัยสุขภาพ : สคร.6 ชลบุรี ชวนปกป้องหัวใจ ในวันหัวใจโลก [อินเตอร์เน็ต]. 2563 [เข้าถึงเมื่อ 14 พ.ค. 2565]. เข้าถึงได้จาก: https://ddc.moph.go.th/odpc6/news.php?news=14902&deptcode=odpc6&news_views=265
3. [ML] Bagging หรือ Boosting คืออะไร ทำงานอย่างไร? [อินเตอร์เน็ต]. [เข้าถึงเมื่อ 14 พ.ค. 2565]. เข้าถึงได้จาก: https://tupleblog.github.io/bagging-boosting/
4. Scikit-learn [อินเตอร์เน็ต]. [เข้าถึงเมื่อ 14 พ.ค. 2565]. เข้าถึงได้จาก: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html
5. การวิเคราะห์ข้อมูลการวิจัยทางวิทยาศาสตร์สุขภาพโดยใช้ การถดถอยลอจีสติก [อินเตอร์เน็ต]. [เข้าถึงเมื่อ 14 พ.ค. 2565]. เข้าถึงได้จาก: http://classroom.takasila.org/classroom/dataupload/takasila/project/29file/Logistic/LogisticSLT.pdf
6. หลักการและการใช้การวิเคราะห์การถดถอยโลจิสติคสำหรับการวิจัย [อินเตอร์เน็ต]. [เข้าถึงเมื่อ 14 พ.ค. 2565]. เข้าถึงได้จาก: http://rdi.rmutsv.ac.th/rmutsvrj/download/year4-issue1-2555/p1.pdf
7. การวิเคราะห์การถดถอยโลจิสติกแบบไบนารีส าหรับการวิจัยทางสังคมศาสตร [อินเตอร์เน็ต]. [เข้าถึงเมื่อ 14 พ.ค. 2565]. เข้าถึงได้จาก: https://so05.tci-thaijo.org/index.php/saujournalssh/article/download/172178/123610/
8. หลักการและการใช้การวิเคราะห์การถดถอยโลจิสติคสำหรับการวิจัย [อินเตอร์เน็ต]. [เข้าถึงเมื่อ 14 พ.ค. 2565]. เข้าถึงได้จาก: https://www.repository.rmutsv.ac.th/bitstream/handle/123456789/1252/FullText.pdf?sequence=1

9. เจาะลึก Random Forest !!!— Part 2 of “รู้จัก Decision Tree, Random Forest, และ XGBoost!!!” [อินเตอร์เน็ต]. [เข้าถึงเมื่อ 14 พ.ค. 2565]. เข้าถึงได้จาก: https://medium.com/@witchapongdaroontham/%E0%B9%80%E0%B8%88%E0%B8%B2%E0%B8%B0%E0%B8%A5%E0%B8%B6%E0%B8%81-random-forest-part-2-of-%E0%B8%A3%E0%B8%B9%E0%B9%89%E0%B8%88%E0%B8%B1%E0%B8%81-decision-tree-random-forest-%E0%B9%81%E0%B8%A5%E0%B8%B0-xgboost-79b9f41a1c1c
10. รู้จัก Decision Tree, Random Forest, และ XGBoost!!! — PART 1 [อินเตอร์เน็ต]. [เข้าถึงเมื่อ 14 พ.ค. 2565]. เข้าถึงได้จาก: https://medium.com/@witchapongdaroontham/%E0%B8%A3%E0%B8%B9%E0%B9%89%E0%B8%88%E0%B8%B1%E0%B8%81-decision-tree-random-forrest-%E0%B9%81%E0%B8%A5%E0%B8%B0-xgboost-part-1-cb49c4ac1315#:~:text=%F0%9F%9A%81%20XGBoost%20%E2%80%94%20Extreme%20Gradient%20Boosting,%E0%B9%80%E0%B8%A3%E0%B8%B5%E0%B8%A2%E0%B8%99%E0%B8%A3%E0%B8%B9%E0%B9%89%E0%B9%80%E0%B8%A1%E0%B8%B7%E0%B9%88%E0%B8%AD%E0%B9%84%E0%B8%A1%E0%B9%88
11. การใช้เทคโนโลยีการเรียนรู้ของเครื่อง เพื่อทำนายผลการเรียนของนักเรียน [อินเตอร์เน็ต]. [เข้าถึงเมื่อ 14 พ.ค. 2565]. เข้าถึงได้จาก: http://ir-ithesis.swu.ac.th/dspace/bitstream/123456789/610/1/gs611130429.pdf
12. Boosting [อินเตอร์เน็ต]. [เข้าถึงเมื่อ 14 พ.ค. 2565]. เข้าถึงได้จาก: https://guopai.github.io/ml-blog11.html
13. Svetlana Ulianova. Cardiovascular Disease dataset [อินเตอร์เน็ต]. 2563 [เข้าถึงเมื่อ 14 พ.ค. 2565]. เข้าถึงได้จาก: https://www.kaggle.com/datasets/sulianova/cardiovascular-disease-dataset
14. Blood pressure chart: What is a healthy range? 8 signs your blood pressure is too high [อินเตอร์เน็ต]. [เข้าถึงเมื่อ 14 พ.ค. 2565]. เข้าถึงได้จาก: https://www.express.co.uk/life-style/health/1454979/blood-pressure-chart-healthy-range-signs-too-high-symptoms-EVG

15. Am I a candidate for bariatric surgery? Learn your body mass index [อินเตอร์เน็ต]. [เข้าถึงเมื่อ 14 พ.ค. 2565]. เข้าถึงได้จาก: https://www.honorhealth.com/medical-services/bariatric-weight-loss-surgery/bariatric-surgery-candidate
16. The Best Guide to Regularization in Machine Learning [อินเตอร์เน็ต]. [เข้าถึงเมื่อ 14 พ.ค. 2565]. เข้าถึงได้จาก: https://www.simplilearn.com/tutorials/machine-learning-tutorial/regularization-in-machine-learning
17. what is C parameter in sklearn Logistic Regression [อินเตอร์เน็ต]. [เข้าถึงเมื่อ 14 พ.ค. 2565]. เข้าถึงได้จาก: https://www.edureka.co/community/164582/what-is-c-parameter-in-sklearn-logistic-regression#:~:text=C%20is%20known%20as%20a,training%20data%20leads%20to%20overfitting.
18. sklearn.linear_model.LogisticRegression [อินเตอร์เน็ต]. [เข้าถึงเมื่อ 14 พ.ค. 2565]. เข้าถึงได้จาก: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
19. sklearn.linear_model.Random Forest Classifier [อินเตอร์เน็ต]. [เข้าถึงเมื่อ 14 พ.ค. 2565]. เข้าถึงได้จาก: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
20. Stackoverflow Understanding max_features parameter in RandomForestRegressor [อินเตอร์เน็ต]. [เข้าถึงเมื่อ 14 พ.ค. 2565]. เข้าถึงได้จาก: https://stackoverflow.com/questions/23939750/understanding-max-features-parameter-in-randomforestregressor#:~:text=%5B%20max_features%20%5D%20is%20the%20size%20of,consider%20when%20splitting%20a%20node.&text=By%20setting%20max_features%20differently%2C%20you,a%20%22true%22%20random%20forest.
21. XGBoost Parameters [อินเตอร์เน็ต]. [เข้าถึงเมื่อ 14 พ.ค. 2565]. เข้าถึงได้จาก: https://xgboost.readthedocs.io/en/stable/parameter.html