การทำนายประสิทธิภาพการทำงานของ JUPYTER NOTEBOOK บน JUPYTERHUB โดยใช้เทคนิคการเรียนรู้ของเครื่อง

PERFORMANCE PREDICTION OF JUPYTER NOTEBOOK IN JUPYTERHUB USING MACHINE LEARNING

ปริวรรต ปธานราษฎร์

บัณฑิตวิทยาลัย มหาวิทยาลัยศรีนครินทรวิโรฒ

2561

การทำนายประสิทธิภาพการทำงานของ JUPYTER NOTEBOOK บน JUPYTERHUB โดยใช้เทคนิคการเรียนรู้ของเครื่อง

ปริวรรต ปธานราษฎร์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร
วิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
คณะวิทยาศาสตร์ มหาวิทยาลัยศรีนครินทรวิโรฒ
ปีการศึกษา 2561

PERFORMANCE PREDICTION OF JUPYTER NOTEBOOK IN JUPYTERHUB USING MACHINE LEARNING

PARIWAT PRATHANRAT

A Thesis Submitted in partial Fulfillment of Requirements

for MASTER OF SCIENCE (Information Technology)

Faculty of Science Srinakharinwirot University

2018

ปริญญานิพนธ์

เรื่อง

การทำนายประสิทธิภาพการทำงานของ JUPYTER NOTEBOOK บน JUPYTERHUB

โดยใช้เทคนิคการเรียนรู้ของเครื่อง

ของ

ปริวรรต ปธานราษฎร์

ได้รับอนุมัติจากบัณฑิตวิทยาลัยให้นับเป็นส่วนหนึ่งของการศึกษาตามหลักสูตร

ปริญญาวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ

ของมหาวิทยาลัยศรีนครินทรวิโรฒ

..... คณบดีบัณฑิตวิทยาลัย

(ผู้ช่วยศาสตราจารย์ นายแพทย์ฉัตรชัย เอกปัญญาสกุล)

.....

คณะกรรมการสอบปากเปล่าปริญญานิพนธ์

..... ที่ปรึกษาหลัก

(ผู้ช่วยศาสตราจารย์ ดร.จันตรี ผลประเสริฐ)

..... ประธาน

(ดร.วีรยุทธ เจริญเรืองกิจ)

..... กรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.ประจักษ์ เฉิดโฉม)

ชื่อเรื่อง	การทำนายประสิทธิภาพการทำงานของ JUPYTER NOTEBOOK บน JUPYTERHUB โดยใช้เทคนิคการเรียนรู้ของเครื่อง
ผู้วิจัย	ปริวรรต ปธานราษฎร์
ปริญญา	วิทยาศาสตรมหาบัณฑิต
ปีการศึกษา	2561
อาจารย์ที่ปรึกษา	ผู้ช่วยศาสตราจารย์ ดร. จันตรี ผลประเสริฐ

ในงานวิจัยนี้ ผู้วิจัยได้ใช้เทคนิคการเรียนรู้ของเครื่องในการทำนายประสิทธิภาพของ Jupyter notebook ที่ทำงานอยู่บน JupyterHub โดยใช้ข้อมูลจาก notebook's CPU profile, notebook's Memory profile, จำนวนผู้ใช้งาน และค่าเฉลี่ยการหน่วงเวลาระหว่าง cell โดยใช้เกณฑ์ชี้วัดเป็น response time สำหรับการวัดความแม่นยำที่ได้จากการทำนายของเทคนิคการเรียนรู้ของเครื่อง ผู้วิจัยจะใช้ mean absolute error (MAE) และ mean absolute percentage error (MAPE) ในการวัดความแม่นยำ

ผลลัพธ์ที่ได้แสดงให้เห็นว่า random forest regression model มีความสามารถในการทำนายประสิทธิภาพของ Jupyter notebook ที่ทำงานอยู่บน JupyterHub ได้ดีโดยค่า MAE ได้ 13.679 วินาที และ MAPE ได้ 9.732% และ KNN regression model ไม่สามารถทำนายภาพสิทธิภาพของ Jupyter notebook ที่ทำงานอยู่บน JupyterHub ได้ โดยค่า MAE ได้ 128.907 วินาทีและ MAPE ได้ 95.057%

คำสำคัญ : Jupyter notebook, JupyterHub, เทคนิคการเรียนรู้ของเครื่อง

Title	PERFORMANCE PREDICTION OF JUPYTER NOTEBOOK IN JUPYTERHUB USING MACHINE LEARNING
Author	PARIWAT PRATHANRAT
Degree	MASTER OF SCIENCE
Academic Year	2018
Thesis Advisor	Chantri Polprasert

In this research, machine learning was employed to predict the performance of the Jupyter notebook on the JupyterHub. We show that the CPU profile of the notebook, Memory profile of the notebook, The number of users and average idle time between cells were crucial features that impacted on the performance of machine learning models to accurately predict the performance of the Jupyter notebook in terms of the response time. The performance of The model was characterized by the prediction of the notebook response time in terms of the mean absolute error (MAE) and mean absolute percentage error (MAPE).

The results releaded that the random forest model yielded the strongest performance to predict the performance of Jupyter notebook with a MAPE equal to 9.732% and a MAE equal to 13.679 seconds and with an r-square equal to 0.93 and the KNN model yielded poorest performance to predict the performance of Jupyter notebook with a MAPE equal to 95.057% and a MAE equal to 128.907 seconds.

Keyword : Jupyter notebook, JupyterHub, Machine Learning

กิตติกรรมประกาศ

ปริญญานิพนธ์นี้สำเร็จลุล่วงด้วยดี ผู้วิจัยขอกราบขอบพระคุณท่านคณาจารย์ทุกท่านทั้งในอดีตและปัจจุบัน ที่ได้กรุณาให้คำแนะนำในการศึกษา และได้รับความช่วยเหลือ และได้รับการแนะนำของผู้ช่วยศาสตราจารย์ ดร. จันตรี ผลประเสริฐ ทั้งในด้านการดำเนินงานวิจัย, ด้านวิชาการ และความช่วยเหลือด้านต่าง ๆ อย่างดีเสมอมา นอกจากนี้ ขอขอบคุณพี่ ๆ เพื่อน ๆ และน้อง ๆ ทุกคนที่เป็นกำลังใจ และให้ความช่วยเหลือในการทำปริญญานิพนธ์นี้

สุดท้ายนี้ ผู้วิจัยขอขอบพระคุณบิดามารดา และครอบครัว ซึ่งเปิดโอกาสให้ได้รับการศึกษาเล่าเรียน ตลอดจนคอยช่วยเหลือและให้กำลังใจผู้วิจัยเสมอมาจนสำเร็จการศึกษา

ปริวรรต ปธานราษฎร์

สารบัญ

หน้า

บทคัดย่อภาษาไทย ..... ง

บทคัดย่อภาษาอังกฤษ ..... จ

กิตติกรรมประกาศ ..... ฉ

สารบัญ ..... ช

สารบัญตาราง ..... ญ

สารบัญรูปภาพ ..... ฎ

บทที่ 1 1

บทนำ 1

1.1 ความสำคัญและที่มาของงานวิจัย 1

1.2 วัตถุประสงค์ของการวิจัย 2

1.3 ขอบเขตของการวิจัย 2

1.4 วิธีการดำเนินการวิจัย 3

1.5 ประโยชน์ที่คาดว่าจะได้รับจากการวิจัย 3

บทที่ 2 แนวคิด ทฤษฎี เทคโนโลยี และระบบงานที่เกี่ยวข้อง 4

2.1 เทคโนโลยีที่เกี่ยวข้อง 4

2.1.1 Jupyter notebook 4

2.1.2 JupyterHub 6

2.1.3 เทคโนโลยีจำลองสภาพแวดล้อมให้กับซอฟแวร์เฉพาะบนเครื่องคอมพิวเตอร์ 7

2.2 ทฤษฎีพื้นฐานเกี่ยวกับ Machine Learning 11

2.2.1 การเรียนรู้ของเครื่อง (Machine Learning) 11

2.2.2 ทฤษฎีเกี่ยวกับ Feature Normalization 12

2.2.3 ทฤษฎีเกี่ยวกับ KNN Regression 13
2.2.4 ทฤษฎีเกี่ยวกับ Random Forest Regression 15
2.3 ทฤษฎีเกี่ยวกับการวัดความแม่นยำของแต่ละโมเดล 18
2.4 งานวิจัยที่เกี่ยวข้อง 18
บทที่ 3 วิธีดำเนินงานวิจัย 20
3.1 โครงสร้างงานวิจัย 20
3.1.1. โครงสร้างระบบ JupyterHub 20
3.1.2. ขั้นตอนการทำงานเทคนิคการเรียนรู้ของเครื่อง 23
3.2 การจัดเตรียมเครื่องมือ 25
3.3 การเก็บรวบรวมข้อมูล (Data collection) 26
3.3.1 การหาไฟล์ notebook สำหรับใช้เก็บข้อมูล 26
3.3.2 การ monitoring และการเก็บข้อมูลของ CPU และ memory 28
3.3.3 การเก็บข้อมูลระยะเวลาที่ใช้ในการประมวลผลของ notebook ด้วย shell script 30
3.4 ตัวแปรที่ใช้สำหรับ Machine Learning Model 32
3.5 Hyper-parameter ที่ใช้สำหรับการทำนายข้อมูล 34
3.5.1 KNN Regression 34
3.5.2 Random Forest Regression 34
บทที่ 4 ผลลัพธ์จากการวิจัย 35
บทที่ 5 สรุปผลการวิจัย อภิปรายผล และข้อเสนอแนะ 41
5.1 สรุปผลการวิจัย 41
5.2 อภิปรายผล 41
5.3 ข้อเสนอแนะ 42
บรรณานุกรม 43
ภาคผนวก 45

ประวัติผู้เขียน 50

สารบัญตาราง

หน้า

ตาราง 1 จำนวน record ของ NB1, NB2, NB3, NBTest 28

ตาราง 2 ตัวอย่างไฟล์ csv ของ CPU ของ container 29

ตาราง 3 ตัวอย่างไฟล์ csv ของ memory ของ container 29

ตาราง 4 รายละเอียด Idle time 32

ตาราง 5 Hyper-parameter ของแต่ละ machine learning model 35

ตาราง 6 ผลลัพธ์การทำนาย response time ของแต่ละ model (วินาที) ส่วนที่ 1 37

ตาราง 7 ผลลัพธ์การทำนาย response time ของแต่ละ model (วินาที) ส่วนที่ 2 38

ตาราง 8 ระยะห่างเฉลี่ยของ NB1, NB2, NB3 กับ NBTest 39

ตาราง 9 วัดความแม่นยำด้วย MAE และ MAPE 40

ตาราง 10 ตัวอย่างรายละเอียดของ NB1, NB2, NB3 และ NBTest 46

สารบัญรูปภาพ

หน้า

ภาพประกอบ 1 Interface การทำงานของ Jupyter notebook 5
ภาพประกอบ 2 หน้าหลักของ Jupyter notebook 5
ภาพประกอบ 3 Flowchart การทำงานของ JupyterHub 6
ภาพประกอบ 4 ความแตกต่างระหว่าง Virtual Machine กับ Container 7
ภาพประกอบ 5 องค์ประกอบการทำงานของ Docker 8
ภาพประกอบ 6 Docker container Dashboard ใน Grafana 9
ภาพประกอบ 7 การเก็บข้อมูลของ Prometheus 10
ภาพประกอบ 8 ประเภทของ Normalization 12
ภาพประกอบ 9 การจับกลุ่มใน KNN 13
ภาพประกอบ 10 รูปแบบการตัดสินใจแบบ Decision tree 15
ภาพประกอบ 11 ตัวอย่างต้น Random Forest แต่ละต้นที่ตัดสินใจออกมาแล้ว เฉลี่ยออกมาเป็นผลลัพธ์ 16
ภาพประกอบ 12 โครงสร้างในการพัฒนางานวิจัย Jupyter notebook 20
ภาพประกอบ 13 หน้า Login เข้าสู่ JupyterHub 21
ภาพประกอบ 14 การตรวจสอบรายละเอียดของ container 22
ภาพประกอบ 15 วิธีการบันทึกไฟล์ในรูปแบบ csv 23
ภาพประกอบ 16 ขั้นตอนการทำงานเทคนิคการเรียนรู้ของเครื่อง 23
ภาพประกอบ 17 การใช้ cross-validate เพื่อหา Hyper-parameter 24
ภาพประกอบ 18 การทำนาย response time ของ test data และการวัดความแม่นยำจากการทำนายด้วย MAE และ MAPE 25
ภาพประกอบ 19 การใช้งาน JupyterHub โหมด admin เพื่อใช้งาน Jupyter notebook container พร้อมกัน 25

ภาพประกอบ 20 กราฟ memory ของ train data (NB1, NB2, NB3)..... 27
ภาพประกอบ 21 กราฟ CPU ของ train data (NB1, NB2, NB3) 27
ภาพประกอบ 22 การสร้างค่า idle time สำหรับใช้หน่วงเวลา 31
ภาพประกอบ 23 การหน่วงเวลาระหว่าง cell 2 cell..... 31
ภาพประกอบ 24 รายละเอียดการแบ่งประเภทระหว่าง idle time และ response time 33
ภาพประกอบ 25 กราฟแสดงการใช้งานของไฟล์ notebook..... 33
ภาพประกอบ 26 กราฟแสดงผลลัพธ์จากการทำนายค่า Response time ของ NBTest โดยใช้ Random forest, KNN และ Baseline 36
ภาพประกอบ 27 Feature Importance ของ Random Forest..... 39
ภาพประกอบ 28 ตัวอย่าง NB1 (ส่วนที่ 1)..... 46
ภาพประกอบ 29 ตัวอย่าง NB1 (ส่วนที่ 2)..... 46
ภาพประกอบ 30 ตัวอย่าง NB2 (ส่วนที่ 1)..... 47
ภาพประกอบ 31 ตัวอย่าง NB2 (ส่วนที่ 2)..... 47
ภาพประกอบ 32 ตัวอย่าง NB3 (ส่วนที่ 1)..... 48
ภาพประกอบ 33 ตัวอย่าง NB3 (ส่วนที่ 2)..... 48
ภาพประกอบ 34 ตัวอย่าง NBTest (ส่วนที่ 1)..... 49
ภาพประกอบ 35 ตัวอย่าง NBTest (ส่วนที่ 2)..... 49

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของงานวิจัย

Jupyter notebook คือ โปรแกรมที่ทำงานอยู่บน browser ใช้สำหรับเขียน programming และมีสามารถในใช้งานด้านเอกสารไปพร้อมกับการเขียน programming ตัวโปรแกรมนั้นมีการสนับสนุนด้านการเขียนโค้ดที่หลากหลาย เช่น ความสามารถในการใช้งานด้านภาษา programming ที่หลากหลาย (Python, R, Java Script, C#) [1], สามารถคำนวณคะแนนการเขียน programming ของนักเรียนโดยใช้ nbgrader [2] เป็นต้น ความสามารถเหล่านี้ทำให้ Jupyter notebook ถูกนำไปใช้ในเรื่องการวิเคราะห์ข้อมูล โดยเฉพาะอย่างยิ่ง ในส่วนของการเรียนการสอน ตัวอย่างเช่น ในมหาวิทยาลัย University of California Berkeley มีผู้สอนหลายคนติดตั้ง Jupyter notebook ลงในคอมพิวเตอร์ของชั้นเรียน เพื่อใช้ในการเรียนการสอน เป็นต้น [3] นอกจากนั้น Jupyter notebook ยังมีส่วนเสริม (add-ons) ที่หลากหลายสำหรับเพิ่มประสิทธิภาพในการทำงาน และ JupyterHub คือหนึ่งในส่วนเสริมของ Jupyter notebook ที่มีประสิทธิภาพ

JupyterHub เป็นส่วนเสริมของ Jupyter notebook ที่คอยจัดการ user ที่เข้ามาใช้งาน Jupyter notebook ภายใน server ที่จัดเตรียมไว้ ทำให้ผู้ใช้ไม่ต้องติดตั้ง Jupyter notebook ที่เครื่องตนเอง สามารถใช้งานที่ไหนก็ได้แค่ติดต่อกับ server ได้ ซึ่ง JupyterHub นั้นสามารถใช้งานได้ทั้งใน public cloud server และ on-premise server [4] ความสามารถเหล่านี้จะช่วยเพิ่มประสิทธิภาพในการใช้งานสำหรับการเรียนการสอนได้ดี แต่เนื่องจากผู้ใช้งานทุกคนต้องเข้าใช้งาน Jupyter notebook ผ่าน server ทุกคน ทำให้ server รองรับภาระการทำงานทั้งหมด ดังนั้นผู้ดูแลระบบ JupyterHub ต้องมีการบริหารจัดการ resource ที่ดีเพื่อให้สามารถรองรับการใช้งานได้อย่างมีประสิทธิภาพที่ดีที่สุด

แนวทางหนึ่งที่เป็นไปได้คือการหาโมเดลการใช้งานของ Jupyter notebook ที่สามารถทำนายประสิทธิภาพของระบบได้อย่างแม่นยำเพื่อให้ผู้ดูแลระบบสามารถคาดคะเนปริมาณการใช้งานได้อย่างเหมาะสม เพื่อให้การบริหารจัดการการใช้งานสามารถทำได้อย่างมีประสิทธิภาพ ยกตัวอย่างเช่น ในการนำ Jupyter notebook มาใช้ในการเรียนการสอน ถ้าผู้ดูแลระบบมีโมเดลการทำนายประสิทธิภาพของ Jupyter notebook ที่จะนำมาใช้ในการเรียนการสอน ผู้ดูแลระบบก็จะคาดเดาได้ว่าในการใช้งาน Jupyter notebook นี้จะต้องใช้ทรัพยากรเท่าไหร่ เวลาเฉลี่ยที่ใช้ในการทำงานจะเป็นอย่างไร ทำให้สามารถบริหารจัดการ resource ได้อย่างมีประสิทธิภาพ แต่อย่างไรก็ตามในปัจจุบันนี้

ยังไม่มีโมเดลที่จะสามารถทำนายประสิทธิภาพการทำงานของ Jupyter notebook บน JupyterHub ใน server ได้อย่างแม่นยำ

ผู้วิจัยจึงเลือกใช้เทคนิคการเรียนรู้ของเครื่อง (Machine Learning) ประเภท supervised learning แบบ regression (KNN, random forest) มาทำนายหาระยะเวลาในการตอบสนองของ server (Response Time) แล้วนำมาวัดผลความแม่นยำจากการทำนายด้วย Mean Absolute Error (MAE) และ Mean Absolute Percentage Error (MAPE) ผลลัพธ์ที่ได้จะนำมาเปรียบเทียบเพื่อหาว่าเทคนิคใดมีความแม่นยำมากที่สุด

งานวิจัยนี้จึงพัฒนาแบบจำลองการทำนายประสิทธิภาพของ Jupyter notebook ที่ทำงานอยู่บน JupyterHub โดยใช้เทคนิคการเรียนรู้ของเครื่อง (Machine Learning) โดยใช้ค่าที่ได้จาก Grafana มาทำนายประสิทธิภาพการใช้งานบน JupyterHub ซึ่งใช้ระยะเวลาในการตอบสนองของระบบ (response time) เป็นตัวชี้วัด

1.2วัตถุประสงค์ของการวิจัย

1. เพื่อศึกษาและเปรียบเทียบการใช้เทคนิคการเรียนรู้ของเครื่องในการทำนายประสิทธิภาพการทำงาน Jupyter notebook บน JupyterHub ในรูปของ response time โดยใช้ข้อมูลของ user behavior และ computer profile

2. เพื่อศึกษา feature ที่ใช้ในโมเดลของเทคนิคการเรียนรู้ของเครื่อง ที่มีผลต่อความแม่นยำในการทำนายการทำงานของ Jupyter notebook บน JupyterHub

1.3 ขอบเขตของการวิจัย

1. ระบบ JupyterHub ในงานวิจัยนี้จะทำงานอยู่บน on-premise server

2. การพัฒนาระบบ JupyterHub บน server จะใช้ Docker Engine ในการสร้าง Jupyter notebook ให้แก่ผู้ใช้งาน ภาษาที่ใช้ในการประมวลผลคือ Python3 เก็บข้อมูลจากโปรแกรม Grafana ที่ดึงข้อมูลมาจาก Prometheus ที่ได้บันทึกการใช้งาน Jupyter notebook ของแต่ละคน โดยข้อมูลที่เก็บจะมีข้อมูล 2 ส่วนคือ user behavior ซึ่งก็คือ idle time มีหน่วยเป็นวินาที และสำหรับ computer profile จะเก็บในส่วนของจำนวนการใช้งาน CPU มีหน่วยเป็น core , จำนวนการใช้งาน memory มีหน่วยเป็น GB และระยะเวลาทั้งหมดของการใช้งาน Jupyter notebook มีหน่วยเป็นวินาที

3.ข้อมูลที่นำมาใช้ นำมาจากการจำลองการใช้งานของผู้ใช้ โดยใช้ script ที่เขียนขึ้นมาเอง ซึ่ง Feature ที่ใช้ในการทำนายข้อมูลมีทั้งหมด 62 Feature ประกอบไปด้วย

- number-of-current-user จำนวน 1 feature
- idle-time จำนวน 1 feature
- AVG-CPU-per-user (ต่อ 10 วินาที) จำนวน 30 feature
- AVG-memory-per-user (ต่อ 10 วินาที) จำนวน 30 feature

4. เทคนิคการเรียนรู้ของเครื่องที่นำมาใช้ทำนาย ผู้วิจัยจะใช้ supervised learning แบบ regression (KNN, random forest) มาใช้ในการทำนาย response และเปรียบเทียบผลลัพธ์ของแต่ละโมเดลด้วย Mean Absolute Error (MAE) และ Mean Absolute Percentage Error (MAPE)

1.4 วิธีการดำเนินการวิจัย

1. ทบทวนวรรณกรรมและงานวิจัย (Literature Review) ที่เกี่ยวข้อง
2. จัดหาและติดตั้งเครื่องมือต่าง ๆ เพื่อใช้ในงานวิจัย
3. เก็บข้อมูลที่ได้จากการวิจัย
4. ปรับแต่งข้อมูลที่ได้มาเพื่อให้พร้อมสำหรับการทำนายข้อมูล
5. ศึกษาวิธีการทำนายข้อมูลโดยใช้ KNN, random forest และ Baseline
6. นำค่าความแม่นยำที่ได้จากการทำนายของเทคนิคการเรียนรู้ของเครื่องมาเปรียบเทียบ
7. ประเมินผลและสรุปงานวิจัย

1.5 ประโยชน์ที่คาดว่าจะได้รับจากการวิจัย

1. เข้าใจการติดตั้งและการใช้งาน JupyterHub บน server
2. เข้าใจกระบวนการใช้งานเทคนิคการเรียนรู้ของเครื่องที่ใช้สำหรับการหาประสิทธิภาพการทำงานของ JupyterHub

บทที่ 2
แนวคิด ทฤษฎี เทคโนโลยี และระบบงานที่เกี่ยวข้อง

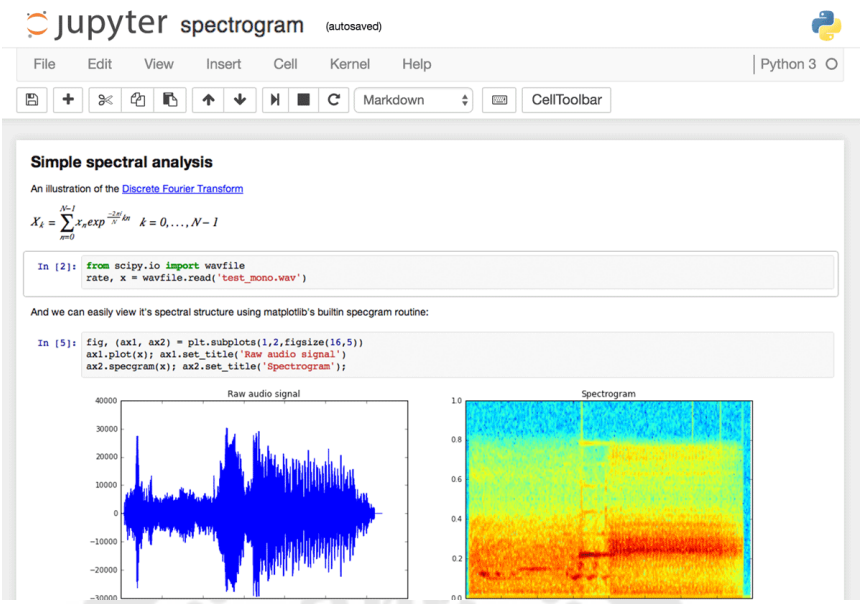
ในการศึกษาของงานวิจัยเรื่อง การทำนายประสิทธิภาพการทำงานของ JUPYTER NOTEBOOK บน JUPYTERHUB โดยใช้เทคนิคการเรียนรู้ของเครื่อง ในครั้งนี้มี 2 ส่วนด้วยกัน ส่วนแรกคือเรื่องเทคโนโลยีที่นำมาใช้สร้างระบบ JupyterHub ต้องทำอย่างไร มีอะไรบ้าง และประเด็นที่สองคือเรื่องเทคนิคการเรียนรู้ของเครื่องซึ่งในการงานวิจัยนี้จะใช้ Random forest regression และ KNN regression ในการทำนาย response time ซึ่งเป็นตัวชี้วัดประสิทธิภาพการทำงานของ server ที่มี JupyterHub ทำงานอยู่

ผู้วิจัยจึงจำเป็นต้องมีความรู้ความเข้าใจ แนวคิด ทฤษฎี เทคโนโลยี และระบบงานที่เกี่ยวข้อง โดยการศึกษาเทคโนโลยีที่เกี่ยวข้อง ทฤษฎีเกี่ยวกับเทคนิคการเรียนรู้ของเครื่องและทฤษฎีการวัดความแม่นยำของแต่ละโมเดลของเทคนิคการเรียนรู้ของเครื่อง เพื่อที่จะสามารถนำไปใช้แก้ปัญหาต่าง ๆ ในงานวิจัยนี้ได้ ซึ่งรายละเอียดต่าง ๆ ประกอบไปด้วย

2.1 เทคโนโลยีที่เกี่ยวข้อง

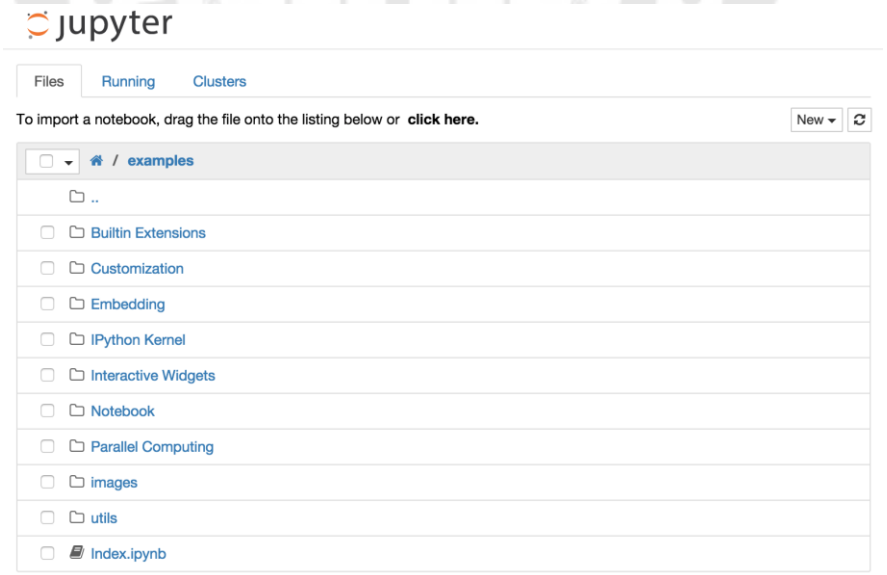
2.1.1 Jupyter notebook

Jupyter notebook เป็น application รูปแบบ server-client ที่ยอมรับการสร้าง แก้ไข และรัน notebook document บน web browser โดย Application รองรับการรันบนคอมพิวเตอร์โดยไม่ต้องใช้อินเตอร์เน็ต สำหรับการใช้งาน Jupyter notebook (ภาพประกอบ 1, 2) นั้น ตัวโปรแกรมจะมี Dashboard ที่ใช้ควบคุมไฟล์เช่น เปลี่ยนชื่อไฟล์ คัดลอกไฟล์ ยอมรับการเปิดและปิด notebook document



ภาพประกอบ 1 Interface การทำงานของ Jupyter notebook

ที่มา: https://www.dataquest.io/blog/images/jupyter/interface-screenshot.png



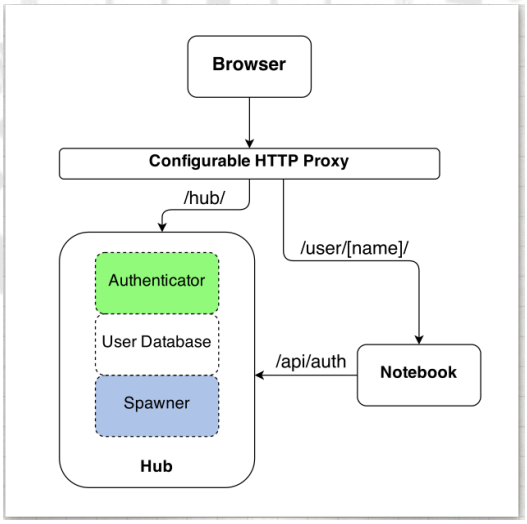
ภาพประกอบ 2 หน้าหลักของ Jupyter notebook

ที่มา: http://jupyter-notebook.readthedocs.io/en/stable/_images/dashboard_files_tab.png

2.1.2 JupyterHub

JupyterHub คือการนำ Jupyter notebook มาใช้ในระดับกลุ่ม โดยการให้ผู้ใช้งานได้มีสิทธิเข้ามาใช้งานในระบบแทนการติดตั้ง Jupyter notebook ลงเครื่องตนเอง กลุ่มผู้ใช้งานโดยเฉพาะอย่างยิ่งในกลุ่มนักเรียน กลุ่มผู้วิจัยและวิเคราะห์ข้อมูล สามารถทำงานร่วมกันได้ภายในกลุ่มที่ผู้ดูแลระบบได้กำหนดไว้ ซึ่ง JupyterHub สามารถติดตั้งได้ทั้งในรูปแบบ on-premise และรูปแบบ on cloud

JupyterHub ที่ถูกติดตั้งลง server จะทำหน้าที่จัดการการใช้งาน Jupyter notebook ให้กับผู้ที่เข้ามาใช้งาน หลักการทำงานคือ เมื่อผู้ใช้งาน login ผ่านเข้ามาผ่าน browser ตัว JupyterHub จะทำการยืนยันตัวตนกับ database หากถูกต้องจะทำการสร้าง notebook มาให้ผู้ใช้งาน (ดังภาพประกอบ 3) และหากผู้ใช้งานเคยเข้ามาใช้งานก่อนหน้านี้ ตัว JupyterHub จะทำการตรวจสอบ backup ของผู้ใช้งานนั้น ถ้าเคยมี ก็จะต่อ backup เข้ากับ notebook ที่สร้างขึ้นทำให้ผู้ใช้ไม่ต้อง backup ลงเครื่องของตนเองเมื่อเลิกใช้งาน และเป็นการแก้ปัญหาที่เกิดจากการใช้เครื่องคอมพิวเตอร์ส่วนตัวได้เพราะสามารถทำงานที่เครื่องไหนก็ได้แค่สามารถติดต่อกับเครื่องที่มี JupyterHub ได้ก็พอ



ภาพประกอบ 3 Flowchart การทำงานของ JupyterHub

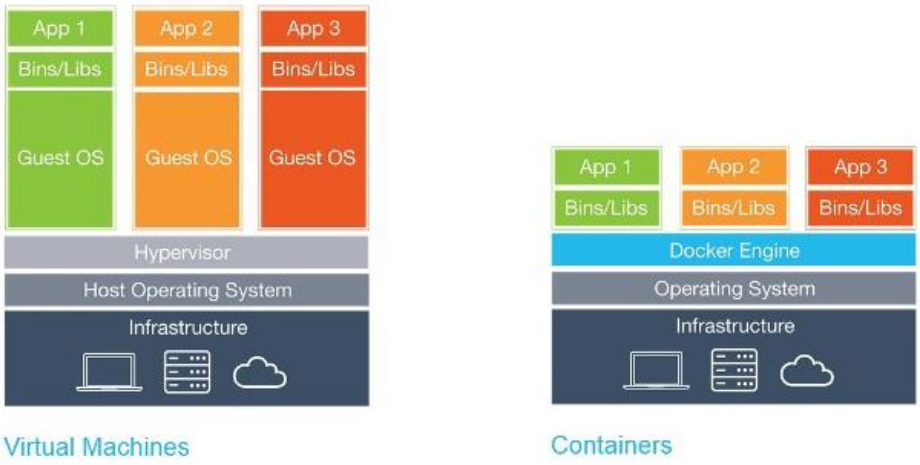
ที่มา: https://jupyterhub.readthedocs.io

จุดเด่นอย่างหนึ่งของ JupyterHub คือการใช้งานร่วมกับ docker ซอฟแวร์คอนเทนเนอร์ที่จะช่วยจำลองสภาพแวดล้อมในกับซอฟแวร์ภายในเครื่อง ทำให้ซอฟแวร์ทำงานได้โดยไม่ไป

รบกวนซอฟแวร์อื่น ๆภายในเครื่องเดียวกัน และยังสามารถส่งซอฟแวร์ที่อยู่ในสภาพแวดล้อมจำลองนั้นไปยังเครื่องอื่นได้โดยที่ใช้ขั้นตอนการติดตั้งไม่เยอะและใช้งานได้ปกติ

2.1.3 เทคโนโลยีจำลองสภาพแวดล้อมให้กับซอฟแวร์เฉพาะบนเครื่องคอมพิวเตอร์

Software Container คือ การสร้างสภาพแวดล้อมเฉพาะให้ซอฟต์แวร์ทำงานได้โดยไม่รบกวนการทำงานของซอฟต์แวร์ตัวอื่นบนระบบปฏิบัติการเดียวกัน เราสามารถนำ Container ไปรันในคอมพิวเตอร์หรือ Server เครื่องไหน ก็สามารถทำงานได้เหมือนเดิมโดยไม่ต้องติดตั้ง resource ของซอฟต์แวร์นั้น โดยในภาพประกอบ 4 จะแสดงถึงความแตกต่างระหว่าง Virtual Machine กับ Container และในงานวิจัย [5, 6] ยังได้เปรียบเทียบความสามารถในการรัน Application ระหว่างทั้งสองระบบด้วย



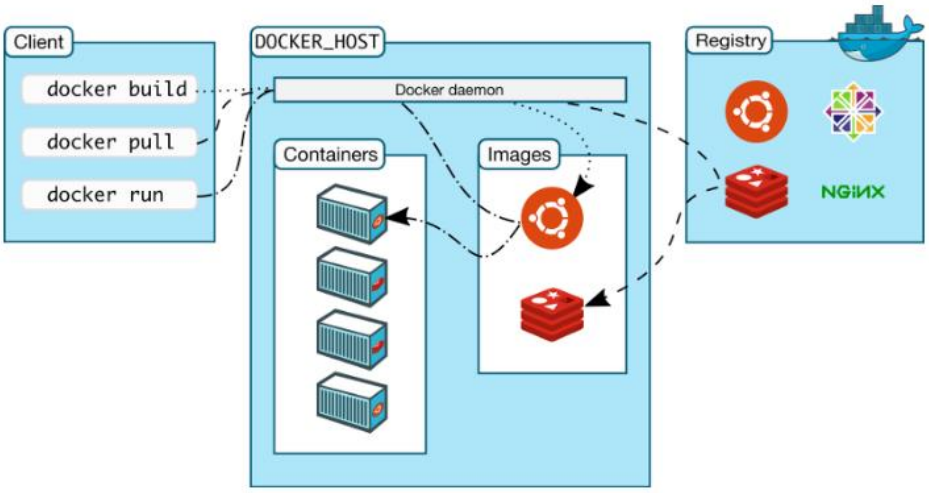
ภาพประกอบ 4 ความแตกต่างระหว่าง Virtual Machine กับ Container

ที่มา: https://medium.com/thothzocial-engineering/ทำความรู้จัก-docker-และ-software-container-c6338629da11

Container จะทำการจำลองและควบคุมสภาพแวดล้อมสำหรับการรันเฉพาะบาง Service เช่น Container ที่รัน nginx ใน ubuntu ก็จะบรรจุ Environment เหล่านี้ไว้เป็น 1 Container และรัน service เท่าที่จำเป็นต้องใช้เท่านั้น ทำให้ใช้ทรัพยากรน้อยกว่า Virtual Machine

ในขณะที่ Virtual Machine จะเป็นการจำลอง Environment มาทั้ง OS รันขึ้นมาเป็นเครื่อง Server 1 เครื่อง และมีการรัน service หลาย ๆ service ใน VM เดียวกัน ทำให้แต่ละ VM ต้องใช้ทรัพยากรจำนวนมาก

2.1.3.1 Docker - เป็น Software Container ที่ใช้จัดการ container โดยการแยกออกเป็นชั้น ๆ โดยใช้แนวคิด build, ship, run (ภาพประกอบ 5) ที่ทำให้ในแต่ละรอบของสร้าง container นั้นจะเกิดขึ้นเร็วมาก และทำให้ระบบ container ได้รับความสนใจอย่างมากในปัจจุบัน ซึ่งในงานวิจัย [7] ได้อธิบายถึงความความสามารถของ docker ในการสร้าง container



ภาพประกอบ 5 องค์ประกอบการทำงานของ Docker

ที่มา: https://medium.com/thothzocial-engineering/ทำความรู้จัก-docker-และ-software-container-c6338629da11

Docker มีองค์ประกอบหลักอยู่ 3 ส่วนคือ

1. Docker image – ต้นแบบของ container ภายในจะมี application ที่ใช้งานและมี resource อื่น ๆ ที่ได้ตั้งค่ามาจาก Dockerfile

2. Docker container – container ที่ถูกสร้างมาจาก image โดยที่ container สามารถใช้งานได้ทันทีที่ติดตั้งเสร็จ

3. Docker registry – เราสามารถนำ Docker Image ไปเก็บบน Public Server (Docker Hub) ได้โดย Docker registry จะเป็นเหมือนตัวเรียกใช้ Docker Image

2.1.3.2 Docker Monitoring - เป็น Monitor สำหรับติดตามการทำงานของ Docker container โดยเฉพาะ ช่วยให้ทราบถึงทรัพยากรที่แต่ละ container ได้ใช้ในขณะนั้น ซึ่ง Grafana [8] (ภาพประกอบ 6) เป็นหนึ่งใน Monitoring Application ที่ใช้รับความนิยมในการเอามาใช้ Monitoring container ของ Docker เช่นกัน

ภาพประกอบ 6 Docker container Dashboard ใน Grafana

ที่มา: https://www.brianchristner.io/content/images/2016/07/Docker_Monitoring_Dashboard-1.png

Grafana ทำหน้าที่เป็น Monitoring resource โดยแบ่งการ monitor ออกเป็น CPU, Memory, Network In, Network Out ให้แต่ละ container โดยรายละเอียดของ Grafanaใน ภาพประกอบ 6 มีดังนี้

1. เกตวัดการใช้งานของ memory, CPU และ Filesystem ในขณะนั้น โดยอยู่ในรูปแบบเปอร์เซ็นต์

2. กราฟการใช้งาน resource ของแต่ละ container แกน x คือ ระยะเวลามีหน่วยเป็นวินาที และแกน y คือ จำนวน resource ที่ถูกใช้งานมีหน่วยเป็นเปอร์เซ็นต์

3. ชื่อของแต่ละ container ที่ทำงานอยู่

4. ค่า resource ที่ถูกใช้ไปของแต่ละ container แสดง 2 หัวข้อคือ AVG ค่า resource เฉลี่ยที่ถูกใช้จาก container นั้น และหัวข้อ current แสดงค่า resource ที่ถูกใช้จาก container ในขณะนั้น

สำหรับผลลัพธ์ที่แสดงบน Grafana นั้นจะดึงข้อมูลมาจาก Prometheus (ภาพประกอบ 7) ที่เป็นเสมือนตัวเก็บค่าทรัพยากรต่าง ๆที่แต่ละ container ใช้ภายในเครื่อง

ภาพประกอบ 7 การเก็บข้อมูลของ Prometheus

ที่มา: https://developers.soundcloud.com/assets/posts/prometheus_bazooka_cpu_top3-e5be5deca9ce640bb6d5e2dd5d243407.png

สำหรับรายละเอียดหน้า Prometheus ในภาพประกอบ 7 มีดังนี้

1. ชื่อหน่วยวัดที่ใช้วัด resource ที่ถูกใช้ในแต่ละ container

2. กราฟแสดง resource ที่ถูกใช้ในแต่ละ container แกน x คือ ระยะเวลามีหน่วยเป็น วินาที และแกน y คือ จำนวน resource ที่ถูกใช้งานมีหน่วยเป็นเปอร์เซ็นต์

3. รายชื่อ container ที่ทำงานอยู่โดยจะมีสีกำกับตรงกับกราฟที่แสดง

2.2 ทฤษฎีพื้นฐานเกี่ยวกับ Machine Learning

สำหรับเทคนิคการเรียนรู้ของเครื่องที่นำมาใช้ในงานวิจัยนั้นจะใช้ทั้งหมด 2 โมเดลคือ random forest regression และ KNN regression และใช้ Feature Normalization ในการจัดการข้อมูลให้พร้อมสำหรับการนำมากับเทคนิคการเรียนรู้ของเครื่อง ซึ่งรายละเอียดต่าง ๆ มีดังนี้

2.2.1 การเรียนรู้ของเครื่อง (Machine Learning)

การเรียนรู้ของเครื่อง เป็นศาสตร์แขนงหนึ่งที่ทำให้คอมพิวเตอร์มีความสามารถในการเรียนรู้ด้วยตนเอง เมื่อรับข้อมูลมา จะสามารถทำนายหรือตัดสินใจได้โดยปราศจากการทำงานตามลำดับคำสั่งโปรแกรม ผลลัพธ์ที่ออกมาขึ้นอยู่กับการเรียนรู้ข้อมูลที่ส่งไปให้คอมพิวเตอร์ โดยผลลัพธ์ที่ได้สามารถนำไปใช้สร้างโมเดลเพื่อใช้ทำนายข้อมูลและช่วยในการตัดสินใจได้ การเรียนรู้ของเครื่องสามารถแบ่งประเภทออกเป็น 3 ประเภทตามหน้าที่และขอบเขตของข้อมูลที่ได้รับคือ supervised learning, unsupervised learning และ reinforcement learning

Supervised learning คือเทคนิคการเรียนรู้ที่ต้องมีคำตอบที่เป็นผลลัพธ์ให้กับโปรแกรมก่อน เพื่อให้โปรแกรมได้เรียนรู้จากตัวอย่างที่ได้มาทำนายผลลัพธ์อีกที ซึ่งการทำนายสามารถแยกออกได้เป็น

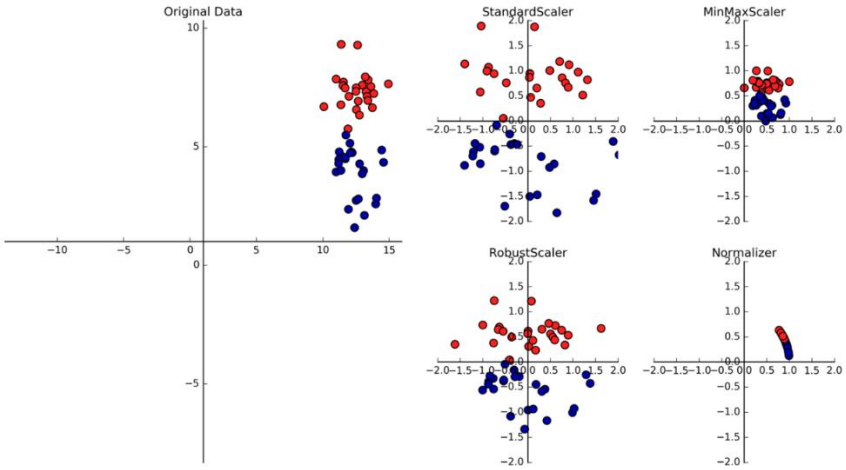
1. Classification – เทคนิคการทำนายที่มีผลลัพธ์เฉพาะ เช่น การทำนายเพศของมนุษย์จะได้ผลลัพธ์สองแบบคือ หนึ่งเป็นผู้ชายและสองเป็นผู้หญิง

2. Regression – เทคนิคการทำนายที่ผลลัพธ์เป็นตัวเลข ผลลัพธ์ที่ได้จากการทำนายจะเป็นรูปแบบอยู่ในช่วงระหว่างของตัวเลข เช่นการทำนายแคลอรีที่ถูกใช้ในแต่ละวันในรอบหนึ่งปี

Unsupervised learning – เทคนิคการเรียนรู้แบบไม่มีผู้สอน โปรแกรมจะหาความสัมพันธ์ของข้อมูลเอง ซึ่งจะใช้การเรียนรู้ในรูปแบบการแบ่งกลุ่ม (clustering) กระบวนการแบ่งกลุ่มข้อมูลนี้เป็นการจัดข้อมูลต่าง ๆ ให้อยู่ในกลุ่มที่เหมาะสม โดยข้อมูลในกลุ่มเดียวกันจะคล้ายกัน และแตกต่างจากข้อมูลในกลุ่มอื่น เช่น การทำนายกลุ่มลูกค้าที่จะเข้ามาใช้งานในมหาลัย

Reinforcement learning – เทคนิคการเรียนรู้แบบเสริมกำลัง เป็นการเรียนรู้แบบสังเกตสิ่งแวดล้อมโดยรอบแล้วกระทำ (action) ให้เกิดผลลัพธ์ที่ดีที่สุด ตัวอย่างเช่น Ai AlphaGo [9] ของ Google เป็น AI สำหรับเล่นโกะ ทุกครั้งที่ผู้เล่นเดินหมาก AlphaGo จะคำนวณและหาวิธีเดินใหม่ตามสิ่งแวดล้อม ซึ่งก็คือการเล่นของผู้เล่นที่เปลี่ยนไป เพื่อหาผลลัพธ์การเดินหมากที่มีโอกาสชนะมากที่สุด

2.2.2 ทฤษฎีเกี่ยวกับ Feature Normalization



ภาพประกอบ 8 ประเภทของ Normalization

ที่มา: https://python-data-science.readthedocs.io/en/latest/normalisation.html

Feature Normalization เป็นเทคนิคการปรับข้อมูลให้อยู่ในระดับเดียวกันเพื่อให้พร้อมสำหรับการนำไปใช้ในเทคนิคการเรียนรู้ของเครื่อง และยังช่วยให้วิเคราะห์ได้เร็วขึ้นและประหยัดทรัพยากรในการใช้ประมวลผล โดยเทคนิคการปรับข้อมูล สามารถทำได้หลายรูปแบบ ดังภาพประกอบ 8 ซึ่งงานวิจัยนี้ได้เลือกใช้ การปรับขนาดข้อมูล (rescaling) ในรูปแบบ StandardScaler ซึ่งเป็นการปรับขนาดข้อมูลให้อยู่ระดับเดียวกัน ผลลัพธ์ของการปรับขนาดข้อมูลจะอยู่ระหว่างค่า 0 และ 1 ดังสมการ (1)

$$z = \frac{x-u}{\sigma} \qquad (1)$$

จากสมการ (1) เมื่อ u คือค่าเฉลี่ยและ σ คือส่วนเบี่ยงเบนค่ามาตรฐานจากค่าเฉลี่ย และ z คือค่ามาตรฐาน สำหรับโมเดล Feature Normalization ที่นำมาใช้ในการงานวิจัยนั้น จะใช้ library ที่มาจาก scikit-learn ซึ่งมีพารามิเตอร์ต่าง ๆ ดังนี้

X: ข้อมูลที่นำมาใช้

axis: มีค่าเป็น 0 และ 1 เป็นแกนกลางที่ใช้คำนวณค่าเฉลี่ยและค่าเบี่ยงเบนมาตรฐาน โดยปกติจะตั้งค่าไว้ที่ 0

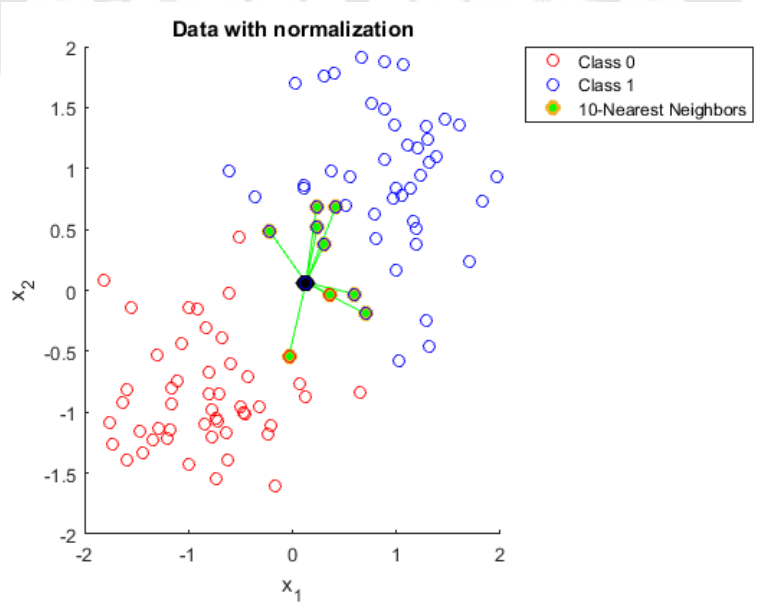
with_mean: มีค่าเป็น True และ False หาก True จะตั้งค่าศูนย์กลางก่อนปรับขนาดข้อมูล

with_std: มีค่าเป็น True และ False หาก True จะตั้งค่าหน่วยเป็น variance

copy: มีค่าเป็น True และ False หาก False จะทำ normalization เรียงเป็นแถวเพื่อหลีกเลี่ยงการคัดลอกข้อมูล

2.2.3 ทฤษฎีเกี่ยวกับ KNN Regression

KNN Regression [10] เป็นการจำลองแบบ supervised learning ที่ใช้ข้อมูลรูปแบบต่อเนื่องเป็นช่วงของตัวเลข โดยใช้ความสัมพันธ์ในลักษณะการจับกลุ่มและตรวจสอบข้อมูลรอบ ๆ หากข้อมูลใดค่าใกล้เคียงกัน จะถูกจับอยู่กลุ่มเดียวกันดังภาพประกอบ 9



ภาพประกอบ 9 การจับกลุ่มใน KNN

ที่ ม า : https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm#/media/File:Map1NNReducedDataSet.png

จากภาพประกอบที่ 9 มีชุดข้อมูล 1 ชุด ซึ่งแบ่งกลุ่มข้อมูลในชุดนั้นออกเป็น 2 กลุ่ม (จุดสีแดงและสีน้ำเงิน) และใช้กลุ่มข้อมูลของทั้งสองกลุ่มที่ใกล้กันเรียกว่า Nearest Neighbors ซึ่งในภาพประกอบมีค่าเท่ากับ 10 (จุดสีเขียว) ผลลัพธ์ที่ได้คือการสร้างข้อมูลตรงกลางของทั้งสองกลุ่มขึ้นมา (จุดสีดำ) ซึ่งเป็นค่าเฉลี่ยของกลุ่มนี้

$$\hat{y} = f(x) = \frac{1}{k}\sum_{j=1}^{k} y_{i_j} \qquad (2)$$

KNN regression จะมีวิธีคำนวณดังสมการ (2) โดยค่าที่ใช้ในการ training จะมี x_i เป็น attribute ที่ใช้หาค่าที่ต้องการทำนาย ส่วน y_i คือค่าจริงที่ต้องการทำนาย และในส่วนของ testing นั้นให้ค่า x คือค่าจริง โดยคำนวณสมการจากการคำนวณระยะทางของ $D(x, x_i)$ ต่อทุกค่าของ training x_i จากนั้นเลือก k ที่ใกล้ที่สุดของทุกค่าของ $x_{i1} \dots x_{ik}$ และใกล้กับค่า $y_{i1} \dots y_{ik}$ ผลลัพธ์ที่ได้คือค่าเฉลี่ยของ $y_{i1} \dots y_{ik}$

สำหรับโมเดลของ KNN regression ในงานวิจัยนี้จะใช้จาก library ของ scikit-learn ซึ่งมีพารามิเตอร์ที่สามารถปรับค่าต่าง ๆเพื่อให้การทำนายแม่นยำมากขึ้นดังนี้

n_neighbors: มีค่าเป็นตัวเลข จำนวนเพื่อนบ้านที่ใช้ในการจับกลุ่ม

weights: รูปแบบการให้น้ำหนักในการทำนาย มีสองรูปแบบคือ Uniform เป็นการให้น้ำหนักของแต่ละ neighborhood เท่ากันหมด และแบบ distance คือการให้น้ำหนักตามระยะห่างจากจุดศูนย์กลาง

algorithm: อัลกอริทึมที่ใช้คำนวณ nearest neighbors

leaf_size: มีค่าเป็นตัวเลข ขนาดการวิเคราะห์ของโมเดลที่เลือกใช้ มีผลในเรื่องความเร็วในการประมวลผลและขนาด memory ที่ใช้

p: พารามิเตอร์สำหรับการวัดระยะห่างโดยหาก p=1 แปลว่าใช้ Minkowski เป็นตัวชี้วัด ซึ่งเทียบเท่ากับการใช้ manhattan_distance (l1) และ euclidean_distance (l2) แต่หาก p=2 แปลว่าจะไม่นำ Minkowski มาใช้เป็นตัวชี้วัด

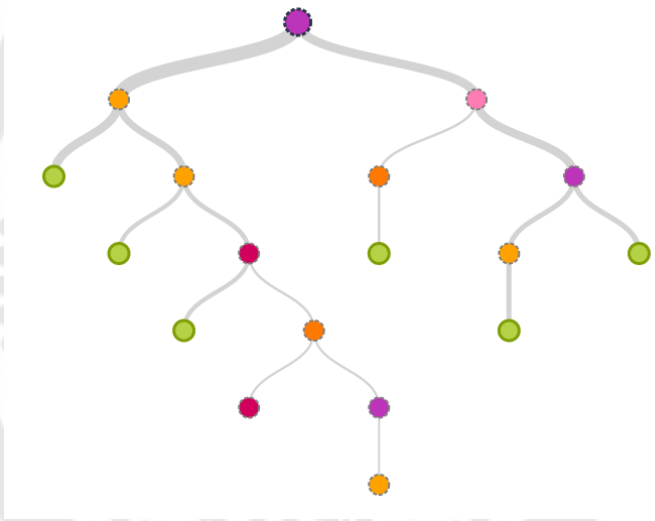
metric: เมตริกที่ใช้วัดระยะทาง ซึ่งโดยปกติจะใช้ Minkowski แต่หากค่า p=2 จะใช้ Euclidean metric ในการวัด

metric_params: เพิ่ม ketwords สำหรับอ้างอิงเหตุผล โดยปกติค่านี้เป็น none

n_jobs: จำนวน processor ที่ใช้ประมวลผลการทำงานพร้อมกัน โดยปกติใช้ 1 แปลว่าใช้ 1 processor ในการประมวลผล แต่หากใช้ -1 แปลว่าใช้ processor ในการประมวลผลทั้งหมด

2.2.4 ทฤษฎีเกี่ยวกับ Random Forest Regression

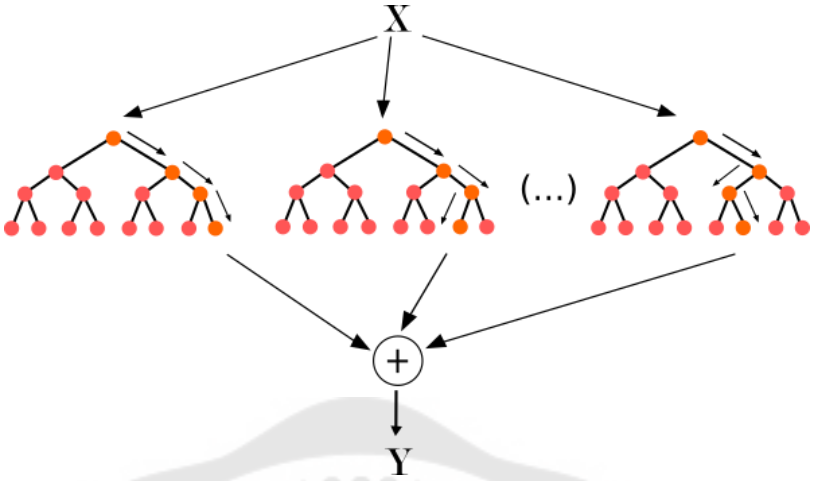
Random Forest Regression [11] เป็นการจำลองแบบ supervised learning ที่มีการทำนายในรูปแบบที่พัฒนามาจาก Decision tree แต่ได้เพิ่มจำนวนต้นขึ้นมา เพื่อช่วยสร้างผลลัพธ์ให้การทำนาย โดย Decision tree นั้นจะทำนายในรูปแบบลำดับชั้น และตัดสินใจแบบ if-else ที่มีคล้ายกับต้นไม้ ดังภาพประกอบ 10



ภาพประกอบ 10 รูปแบบการตัดสินใจแบบ Decision tree

ที่ ม า : https://cdn-images-1 .medium.com/max/1 6 0 0 / 1 * h2 2 XDY1 LDFYkuMvT AjnZtA.png

แต่ Random forest นั้นจะเพิ่มจำนวนต้นขึ้นมาและแต่ละต้นเป็นอิสระต่อกัน โดยใช้ชุดข้อมูลเรียนรู้ย่อยที่ได้จากการสุ่มขึ้นมาใช้ในการทดสอบซึ่งอาจมีการสุ่มที่ซ้ำ ซึ่งต่างจาก Decision tree ที่แต่ละต้นจะสุ่มขึ้นมาจากตัวแปรที่มีทำให้ tree ไม่เหมือนกัน โดยในการทำนายข้อมูลหนึ่งนั้นจะใช้ต้นทุก ๆ ต้นตัดสินใจพร้อมกันและเฉลี่ยความน่าจะเป็นออกมาดังภาพประกอบ 11



ภาพประกอบ 11 ตัวอย่างต้น Random Forest แต่ละต้นที่ตัดสินใจออกมาแล้วเฉลี่ยออกมาเป็นผลลัพธ์

ที่มา: https://cdn-images-1.medium.com/max/1600/1*V1gaYbSzecaGE7s6WVlcsO.png

โมเดลของ Random Forest regression ในงานวิจัยนี้จะใช้จาก library ของ scikit-learn ซึ่งมีพารามิเตอร์ที่สามารถปรับค่าต่าง ๆเพื่อให้การทำนายแม่นยำมากขึ้นดังนี้

n_estimators: มีค่าเป็นตัวเลข จำนวนต้นไม้ทั้งหมดในการทำนาย ยิ่งค่ามากยิ่งแม่นยำและยิ่งประมวลผลนาน

criterion: ฟังก์ชันการวัดคุณภาพการแบ่ง โดยพื้นฐานจะใช้ mean squared error (mse)

max_depth: ความลึกสูงสุดของต้นไม้ ถ้าไม่ได้ตั้งค่าไว้ node จะขยายไปเรื่อย ๆ จนครบใบที่มีทั้งหมด

min_samples_split: จำนวนขั้นต่ำที่ใช้ในการแบ่ง node

min_samples_leaf: จำนวนขั้นต่ำของ node ใบ ซึ่งจะอยู่ในทุก ๆ จุดแยกของ node โดยพื้นฐานคือ 1

min_weight_fraction_leaf: เศษส่วนขั้นต่ำของการให้น้ำหนัก (weight) ทั้งหมด (จากตัวอย่างข้อมูลทั้งหมด) โดยพื้นฐานจะสั่งค่าไว้ที่ 0

max_features: จำนวน features ในการ split ที่ดีที่สุด โดยแบ่งเป็น “auto” คือ จำนวน features สูงสุดจะเท่ากับจำนวน features ทั้งหมด, “sqrt” คือจำนวน features สูงสุดจะเท่ากับ sqrt(จำนวน features), “log2” คือจำนวน features สูงสุดจะเท่ากับ log2(จำนวน features) ซึ่งพื้นฐานที่ตั้งค่าไว้คือ “auto”

max_leaf_nodes: จำกัดการสร้างต้นไม้ด้วย node ใบที่สูงและดีที่สุด โดย node ที่ดีที่สุดหมายถึง การมีความสัมพันธ์ที่แย่น้อยที่สุด ซึ่งโดยพื้นฐานจะตั้งค่าไว้ที่ none คือไม่จำกัด

min_impurity_decrease: node ที่ถูกแบ่งเพื่อลดค่าที่มีความสัมพันธ์แย่จะกำหนดด้วยค่านี้ โดยปกติค่านี้จะมีค่าเท่ากับ 0

min_impurity_split: เกณฑ์การกำหนดการสร้างต้น โดย node จะแยกออกมาเมื่อค่าความสัมพันธ์แย่เกินเกณฑ์ที่กำหนด โดยพื้นฐานค่านี้จะอยู่ที่ 1e-7

bootstrap: ใช้หรือไม่ใช้ bootstrap ในการสร้างต้น โดยปกติจะตั้งค่าไว้ที่ True

oob_score: ใช้หรือไม่ใช้ค่า out-of-bag ในการประเมิณค่าที่มองไม่เห็น โดยพื้นฐานจะตั้งค่าไว้ที่ False

n_jobs: จำนวน processor ที่ใช้ประมวลผลการทำงานพร้อมกัน โดยปกติใช้ 1 แปลว่าใช้ 1 processor ในการประมวลผล แต่หากใช้ -1 แปลว่าใช้ processor ในการประมวลผลทั้งหมด

random_states: ค่าให้การสุ่มตั้งต้น โดยปกติค่านี้จะสุ่มออกมาทุกครั้ง เราสามารถตั้งค่านี้ได้ เพื่อให้ค่าที่ทำนายคงที่

verbose: ค่าที่ใช้ควบคุม verbosity เมื่อทำการ fit และทำนายข้อมูล โดยพื้นฐานจะตั้งค่าที่ 0

warm_start: ค่าที่ใช้ในการเรียก fit ข้อมูล และเพิ่ม estimators โดยพื้นฐานจะตั้งค่าเป็น False

2.3 ทฤษฎีเกี่ยวกับการวัดความแม่นยำของแต่ละโมเดล

ในงานวิจัยนี้ตัวชี้วัดที่จะใช้ในการวัดประสิทธิผลของการทำนายจะใช้ทั้งหมด 2 ค่า ดังนี้

1. ค่าเฉลี่ยความคลาดเคลื่อนสัมบูรณ์ (Mean Absolute Error: MAE) คือการหาค่าสัมบูรณ์ของค่าความแตกต่างระหว่างค่าจริงและค่าที่ทำนายได้

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|x_i - \widehat{x_i}| \qquad (3)$$

ในสมการ (3) x_i คือค่าจริงและ $\widehat{x_i}$ คือค่าที่ทำนาย และ n คือจำนวนข้อมูลที่ใช้ในการทดสอบ

2. ร้อยละของค่าความคลาดเคลื่อนสัมบูรณ์ (Mean Absolute Percentage Error: MAPE) เป็นค่าร้อยละ การวัดความถูกต้องของการทำนาย ซึ่งวัดได้จากค่าความคลาดเคลื่อนของการทำนายได้เทียบกับค่าจริง

$$MAPE = \frac{100}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{|y_i|} \qquad (4)$$

ในสมการ (4) y_i คือค่าจริง $\hat{y}_i$ คือค่าที่ทำนาย และ n คือจำนวนข้อมูลที่ใช้ในการทดสอบ

2.4 งานวิจัยที่เกี่ยวข้อง

งานวิจัย [12] นำเสนอเกี่ยวกับการใช้ Machine learning ในการทำนายประสิทธิภาพของระบบ TOB (Total Order Broadcast) ซึ่งเป็นระบบที่ใช้สำหรับจำลองการพัฒนา application งานวิจัยนี้ได้ใช้ Machine learning ในการทำนาย Latency ภายใน Networks ซึ่งการใช้ Machine learning แบบนี้มีรูปแบบคล้ายกับงานของผู้วิจัย แต่งานวิจัยนี้ลงรายละเอียดเกี่ยวกับ Network ต่างจากงานของผู้วิจัยที่ลงรายละเอียดเกี่ยวกับประสิทธิภาพการทำงานภายในเครื่อง server

ในงานวิจัย [13] จะนำ analytical model มาช่วยเพิ่มประสิทธิภาพในการใช้ machine learning ในการทำนายระยะเวลาที่ server ตอบสนอง (response time) โดยการใช้ analytical model มาสร้าง training set เพื่อให้ machine learning นำไปใช้ทำนาย และเปรียบเทียบ

training set ที่ได้จาก analytical model ไปเรื่อย ๆ เพื่อหา training set ที่ทำให้ค่า machine learning ออกมาดีที่สุด

งานวิจัย [14] นำเสนอการใช้ Machine learning model ในการทำนายการเพิ่มจำนวน Pod ที่บริหารจัดการด้วย Kubernetes โดยงานวิจัยแสดงให้เห็นว่าระบบที่นำเสนอสามารถทำนายการเพิ่มของ Pod โดยพิจารณาจาก %CPU load ได้อย่างแม่นยำและลดการ response time ของระบบเมื่อเทียบกับวิธีดังเดิม ซึ่งจะทำนายการเพิ่ม Pod แบบ threshold-based ได้อย่างมีประสิทธิภาพภายใต้ traffic pattern หลาย ๆ รูปแบบ

งานวิจัย [15] นำเสนอการใช้ Machine Learning model ในการทำนาย response time ของระบบโดยถ้า response time ของระบบมีแนวโน้มที่จะมีค่ามากเกินกว่าค่าที่ยอมรับได้ ระบบจะทำการเพิ่มจำนวน Pod เพื่อรองรับ load ที่เข้ามา feature ที่ใช้ในการทำนายจะประกอบไปด้วย CPU, memory, network throughput โดยใช้โมเดล Random Forest, Gradient Boosting และ Linear regression ในการทำนาย ผลลัพธ์ในการทำนายจะแสดงในรูป MAE (Mean Absolute Error) และ MSE (mean squared error) ซึ่งจากผลการทดลองพบว่า Random Forest มีความแม่นยำในการทำนายมากที่สุด

บทที่ 3
วิธีดำเนินงานวิจัย

ขั้นตอนในการดำเนินงานวิจัยนั้นจะแบ่งออกเป็น 2 ส่วนคือ ส่วนการพัฒนาระบบ JupyterHub และส่วนการใช้เทคนิคการเรียนรู้ของเครื่องเพื่อใช้ทำนายประสิทธิภาพการทำงานของ Jupyter notebook ที่ทำงานอยู่บน JupyterHub ซึ่งมีขั้นตอนการดำเนินงานวิจัยดังนี้

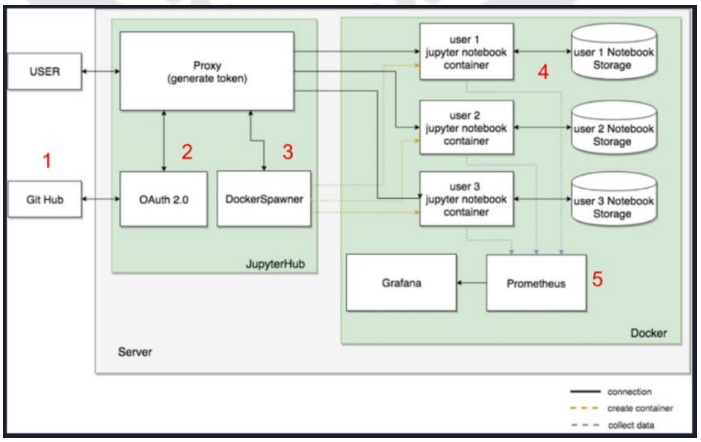
1. การออกแบบโครงสร้างงานวิจัย
2. การจัดเตรียมเครื่องมือ
3. การเก็บรวบรวมข้อมูล
4. การเลือกตัวแปรที่ใช้สำหรับ Machine Learning Model
5. Hyper-parameter ที่ใช้สำหรับการทำนายข้อมูล

3.1 โครงสร้างงานวิจัย

สำหรับโครงสร้างงานวิจัยจะถูกแบ่งออกมา 2 ส่วนคือ โครงสร้างระบบ JupyterHub และขั้นตอนการทำงานเทคนิคการเรียนรู้ของเครื่องในงานวิจัยนี้

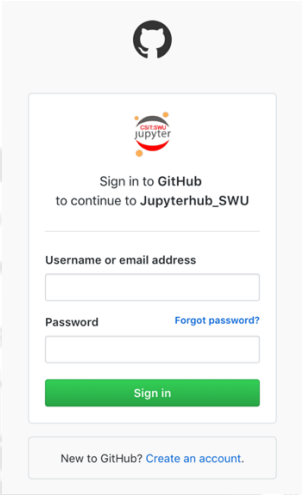
3.1.1. โครงสร้างระบบ JupyterHub

ผู้วิจัยได้ทำการออกแบบและสร้างโครงสร้าง JupyterHub ตามภาพประกอบ 12 ซึ่งโครงสร้างระบบทั้งหมดนี้อยู่ภายใน server ตัวเดียวกัน โดยมีขั้นตอนการทำงาน ดังนี้



ภาพประกอบ 12 โครงสร้างในการพัฒนางานวิจัย Jupyter notebook

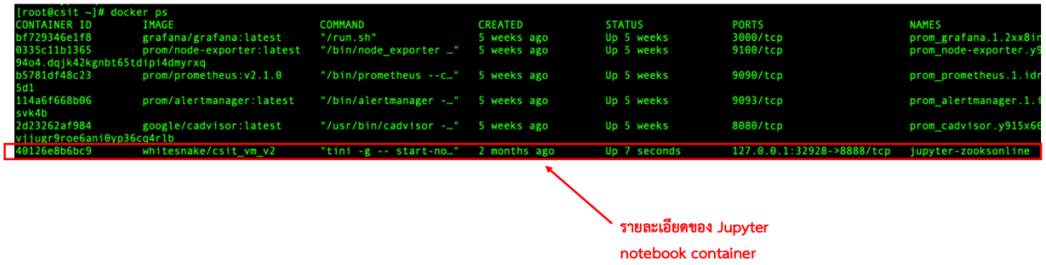
1. เริ่มจาก user เข้าใช้งานจากคอมพิวเตอร์ส่วนบุคคลที่ใดก็ได้ที่อยู่ภายในเครือข่ายที่สามารถ login เข้ามาใช้งานที่ server ได้ ดังภาพประกอบ 13 โดยที่ server จะตรวจสอบสิทธิการเข้าถึงด้วยวิธี OAuth 2.0 ที่ผู้ใช้งานจะต้องมี user จาก GitHub และได้รับการยอมรับสร้าง server หากผู้ใช้งานไม่มี user GitHub จะต้องไปสมัครที่เว็บไซต์ของ GitHub และขอสิทธิ์จากผู้ดูแล server เพื่อเข้าใช้งาน



ภาพประกอบ 13 หน้า Login เข้าสู่ JupyterHub

2. เมื่อ login เข้ามาแล้วทาง JupyterHub ที่อยู่บน server จะทำการตรวจสอบว่า user ที่เข้ามาใช้งานนั้นเคยมีการเข้ามาใช้งานแล้วหรือยัง หากยัง JupyterHub จะใช้ DockerSpawner ทำการสร้าง Jupyter notebook container ลงใน Docker และจัดพื้นที่ภายใน server เพื่อให้สามารถเก็บข้อมูลและข้อมูลไม่หายไปเวลา log out ออก

3. ในกรณีที่ตรวจสอบแล้วพบว่า user ที่เข้ามาใช้งานนั้นเคยมีการเข้ามาใช้งานแล้ว JupyterHub จะใช้ DockerSpawner ทำการ Start Jupyter notebook container ที่มีอยู่และเชื่อมต่อ Jupyter notebook container และข้อมูลที่เก็บไว้เข้าด้วยกัน ดังภาพประกอบ 14 ที่แสดงในเห็นในส่วนของระบบ docker ตรง container ที่ชื่อว่า jupyter-ชื่อผู้ใช้ ที่เพิ่งเชื่อมต่อเข้ามาในระบบ (ในรูปคือเมื่อ 7 วินาทีก่อน) จะ start container ที่มีอยู่ โดยไม่ต้องสร้าง container ใหม่ ซึ่งผู้ดูแลสามารถตรวจสอบการทำงานของระบบได้ด้วยคำสั่ง docker ps



ภาพประกอบ 14 การตรวจสอบรายละเอียดของ container

สำหรับรายละเอียดของ Container ในคำสั่ง docker ps มีดังนี้

Container ID – ID บ่งบอก container ซึ่งจะไม่ซ้ำกันในเครื่องเดียวกัน

Image – แม่พิมพ์ที่ container นั้นใช้

Command – คำสั่งที่รัน container

Created – สร้าง container มานานเท่าไร

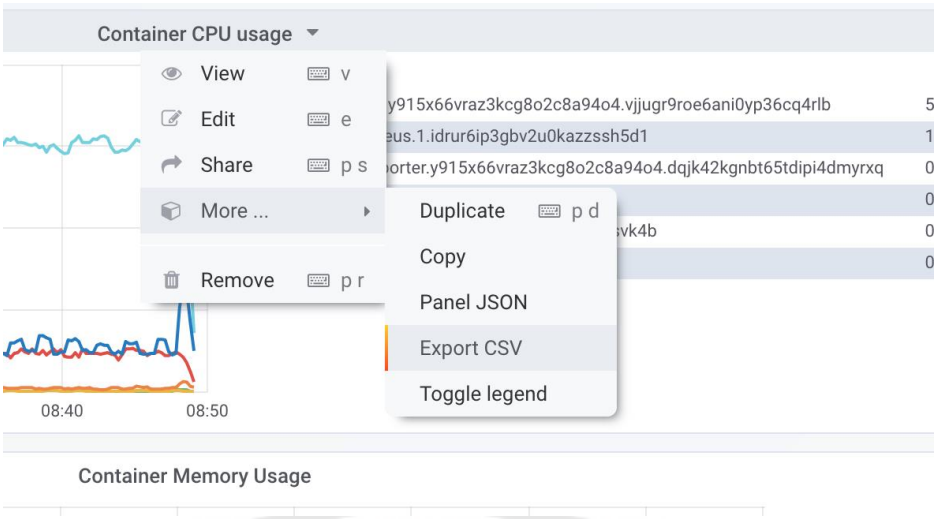
Status – container นี้ทำงานติดต่อกันมานานแค่ไหน

Port – ช่องทางการติดต่อที่ container นี้ใช้

Names – ชื่อ container

4. ในช่วงที่ผู้ใช้งานกำลังใช้งาน notebook อยู่นั้น จะมี Prometheus คอยเก็บข้อมูลการใช้งานอยู่ตลอด โดยจะเก็บทั้ง CPU usage, memory usage, network input, network output

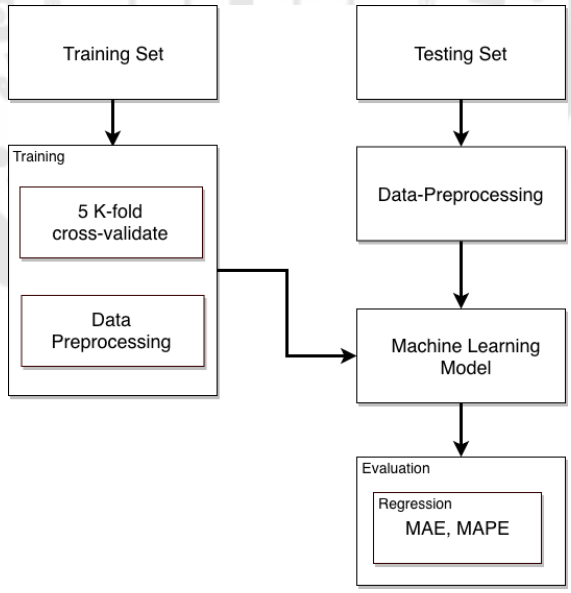
5. ค่าต่าง ๆ ที่ถูกเก็บเข้ามาได้นั้นจะถูกแสดงออกมาด้วย Grafana ในลักษณะของกราฟและทางผู้ดูแลสามารถดึงข้อมูลในลักษณะ csv ดังภาพประกอบ 15 เพื่อใช้ในการทำเทคนิคการเรียนรู้ของเครื่อง (Machine Learning) ต่อไป



ภาพประกอบ 15 วิธีการบันทึกไฟล์ในรูปแบบ csv

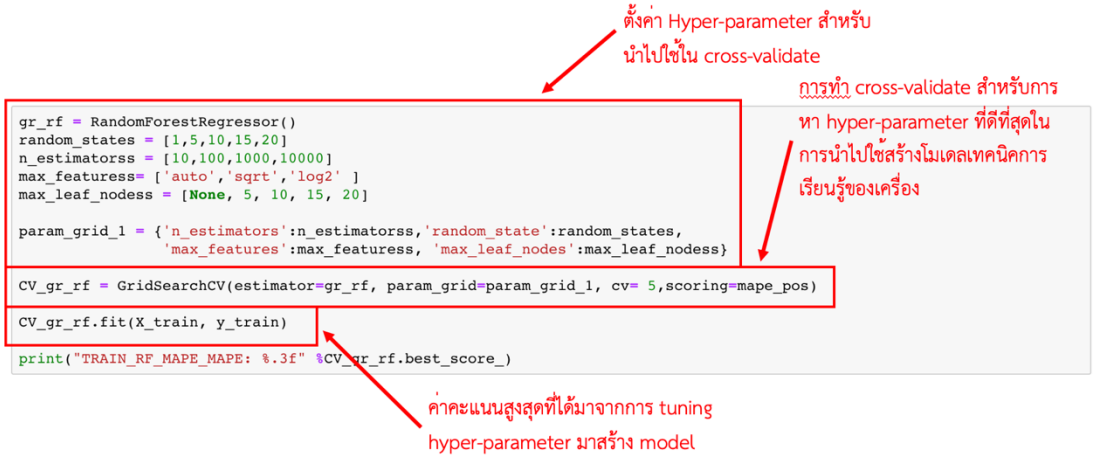
3.1.2. ขั้นตอนการทำงานเทคนิคการเรียนรู้ของเครื่อง

จากภาพประกอบ 16 จะแสดงขั้นตอนการทำเทคนิคการเรียนรู้ของเครื่องในงานวิจัยนี้ ดังนี้



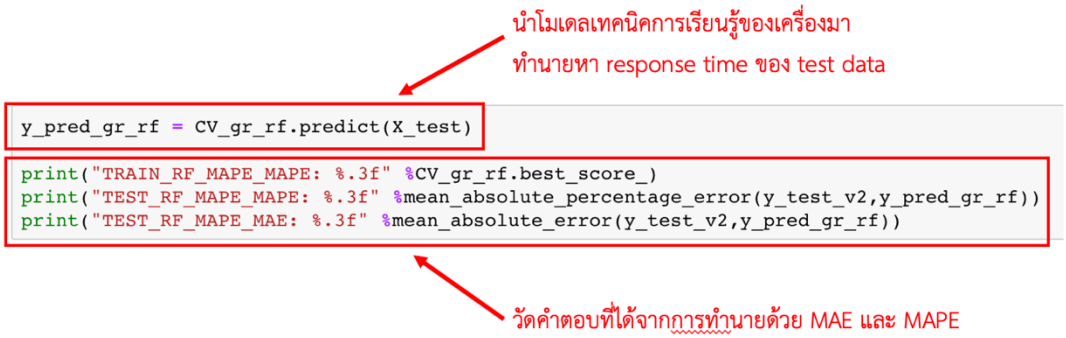
ภาพประกอบ 16 ขั้นตอนการทำงานเทคนิคการเรียนรู้ของเครื่อง

1. เริ่มจากการนำข้อมูลการทำงานของ CPU และ memory ที่เก็บจาก Grafana และ time ที่เก็บโดย script ที่เขียนไว้มารวมกันและแบ่งประเภท feature ออกมาเพื่อใช้สำหรับทำเทคนิคการเรียนรู้ของเครื่อง จากนั้นจึงนำข้อมูลที่ได้มาทำ normalization ในรูปแบบ StandardScaler เพื่อให้ข้อมูลอยู่ในระดับเดียวกันเพื่อเพิ่มประสิทธิภาพความแม่นยำในการทำนาย จากนั้นจึงเข้ากระบวนการ cross-validate ดังภาพประกอบ 17 เพื่อหาค่า Hyper-parameter ที่ใช้สำหรับทำเทคนิคการเรียนรู้ของเครื่อง (Machine Learning) และนำค่าที่ได้มาสร้าง model เทคนิคการเรียนรู้ของเครื่อง



ภาพประกอบ 17 การใช้ cross-validate เพื่อหา Hyper-parameter

2. หลังจากได้ model แล้วจึงการทำนายข้อมูลโดยใช้ test data ดังภาพประกอบ 18 โดย test data ที่ใช้ในภาพประกอบ 18 คือ X_test หลังจากนั้นจึงทำการทดสอบความแม่นยำโดยใช้ MAE และ MAPE เป็นตัวชี้ความแม่นยำ



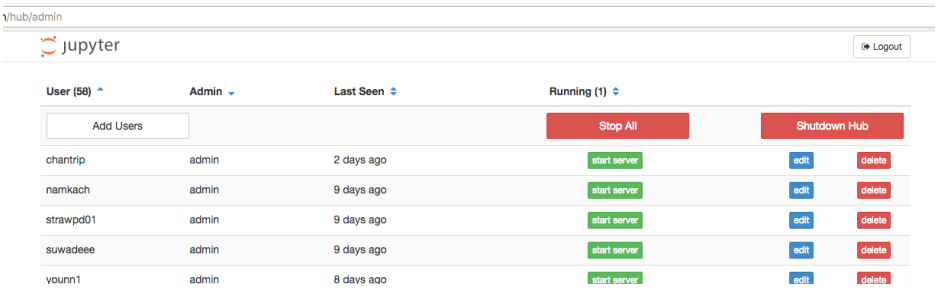
ภาพประกอบ 18 การทำนาย response time ของ test data และการวัดความแม่นยำจากการทำนายด้วย MAE และ MAPE

3. ตรวจสอบและเปรียบเทียบผลลัพธ์ที่ได้และดูว่าเทคนิคไหนดีที่สุด

3.2 การจัดเตรียมเครื่องมือ

1. Server – ผู้วิจัยจะทำการติดตั้งระบบปฏิบัติการ Linux โดยคุณสมบัติของ server ที่นำมาใช้งานจะประกอบด้วย CPU (รวม Threads) ทั้งหมด 32 core และมี RAM (memory) ทั้งหมด 128 GB และ Hard disk ประเภท SAS ทั้งหมด 3 TB

2. JupyterHub (ภาพประกอบ 19) – ทางผู้วิจัยจะทำการติดตั้ง JupyterHub ลงในเครื่อง เนื่องจากการเก็บข้อมูลจำเป็นจะต้องใช้ Jupyter notebook จำนวนมากรันอยู่บน server เครื่องเดียว จึงต้องมี JupyterHub คอยจัดการการเข้าใช้งานและการสร้าง instance Jupyter notebook เพื่อรองรับการใช้งาน โดย JupyterHub จะรันอยู่ในเครื่อง server และสร้าง Jupyter notebook อยู่ใน container ของ docker



ภาพประกอบ 19 การใช้งาน JupyterHub โหมด admin เพื่อใช้งาน Jupyter notebook container พร้อมกัน

3. Jupyter notebook Dockerfile – การใช้งาน Jupyter notebook ในงานวิจัยนี้จำเป็นจะต้องมี library เฉพาะสำหรับการรันข้อมูลที่มี ซึ่งผู้วิจัยได้ใช้พื้นฐานมาจาก SciPy - notebook แหละนำมาปรับแต่งเพิ่มเพื่อให้รองรับการทำวิจัยนี้

4. Docker – ทำหน้าที่รองรับการสร้าง Jupyter notebook จาก JupyterHub และการ monitoring คอยเก็บของ resource ที่ถูกใช้จากการใช้งานบน container

5. Prometheus และ Grafana – monitor ที่จะทำหน้าที่บันทึกและแสดงผลการใช้งาน resource บนเครื่อง server โดย Prometheus จะทำหน้าที่เก็บข้อมูลและ Grafana จะทำการแสดงผลข้อมูลที่ถูกบันทึกไว้

6. notebook file ที่ถูกปรับแต่งโค้ดภายในเพื่อการเก็บข้อมูล – notebook ที่ผู้วิจัยนำมาใช้จะต้องถูกปรับแต่งโค้ดบางส่วนเพื่อให้สามารถดึงข้อมูลสำหรับการทำวิจัยได้

7. Shell Script สำหรับการรวบรวมข้อมูล – เนื่องจากผู้วิจัยต้องเก็บข้อมูลจำนวนมากมาใช้ในการทำ machine learning จึงจำเป็นต้องใช้ script สำหรับการเก็บข้อมูล เพื่อให้การเก็บข้อมูลให้เร็วยิ่งขึ้น

3.3 การเก็บรวบรวมข้อมูล (Data collection)

3.3.1 การหาไฟล์ notebook สำหรับใช้เก็บข้อมูล

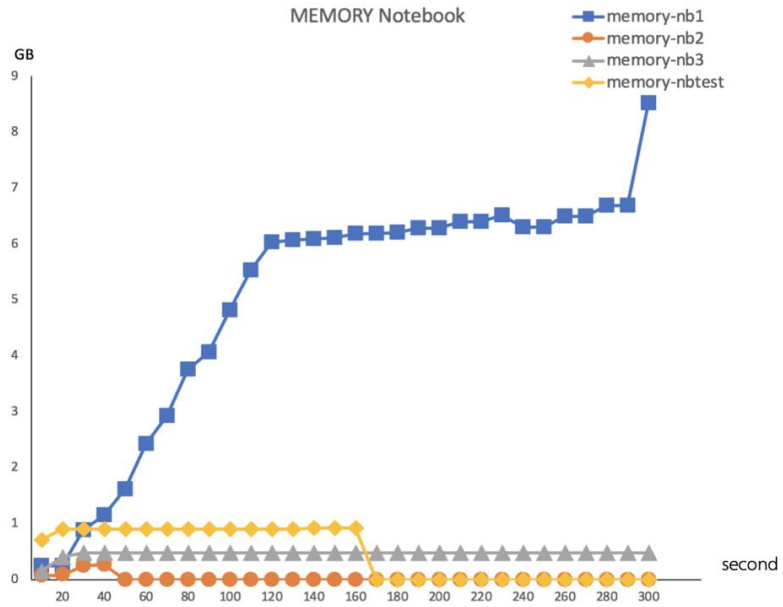
คุณสมบัติของไฟล์ notebook ที่นำมาใช้มีทั้งหมด 4 ไฟล์ แบ่งออกเป็น 3 ไฟล์สำหรับการ Train data และ 1 ไฟล์ สำหรับ Test data โดยมีคุณสมบัติต่าง ๆ ดังนี้

1. NB1 - เป็นไฟล์สำหรับ Train รูปแบบการทำงานจะใช้ CPU ในการประมวลผลมาก แต่ใช้ memory ในการประมวลผลน้อย และใช้ระยะเวลาในการประมวลผลนาน มีค่า response time เฉลี่ยอยู่ที่ 292.58 วินาที

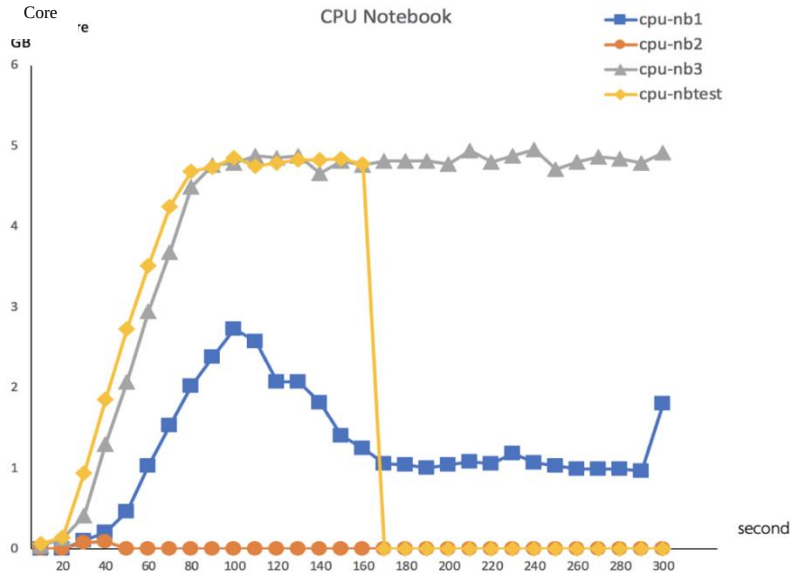
2. NB2 - เป็นไฟล์สำหรับ Train รูปแบบการทำงานจะใช้ทั้ง CPU และ memory ในการประมวลผลน้อย และใช้ระยะเวลาในการประมวลผลสั้น มีค่า response time เฉลี่ยอยู่ที่ 7.33 วินาที

3. NB3 - เป็นไฟล์สำหรับ Train รูปแบบการทำงานจะใช้ memory ในการประมวลผลมาก แต่ใช้ CPU ในการประมวลผลน้อย และใช้ระยะเวลาในการประมวลผลนาน มีค่า response time เฉลี่ยอยู่ที่ 252.80 วินาที

4. NBTest - เป็นไฟล์สำหรับ Test รูปแบบการทำงานจะใช้ memory ในการประมวลผลมาก แต่ใช้ CPU ในการประมวลผลน้อย และใช้ระยะเวลาในการประมวลผลสั้น มีค่า response time เฉลี่ยอยู่ที่ 135.99 วินาที



ภาพประกอบ 20 กราฟ memory ของ train data (NB1, NB2, NB3)



ภาพประกอบ 21 กราฟ CPU ของ train data (NB1, NB2, NB3)

เมื่อนำข้อมูลของทั้งสามไฟล์มาแสดงเป็นกราฟจะได้ดังภาพประกอบ 20 ซึ่งเป็นรายละเอียดการใช้งาน memory ของทั้งสี่ไฟล์ โดยแกน Y คือจำนวน resource ของ memory ที่ถูกใช้งาน มีหน่วยเป็น GB และแกน X คือช่วงเวลามีหน่วยเป็นวินาที และภาพประกอบ 21 ซึ่งเป็นรายละเอียดการใช้งาน CPU ของทั้งสี่ไฟล์ โดยแกน Y คือจำนวน resource ของ CPU ที่ถูกใช้งาน มีหน่วยเป็น core และแกน X คือช่วงเวลามีหน่วยเป็นวินาที

สำหรับจำนวน records ที่ถูกบันทึกนั้นจะมีทั้งหมด 993 records แบ่งเป็น NB1 จำนวน 289 records, NB2 จำนวน 557 records, NB1 จำนวน 63 records และ NB1 จำนวน 84 records ดังตาราง 1

ตาราง 1 จำนวน record ของ NB1, NB2, NB3, NBTest

ชื่อไฟล์	**NB1**	**NB2**	**NB3**	**NBTest**
จำนวนข้อมูล(records)	289	557	63	84

3.3.2 การ monitoring และการเก็บข้อมูลของ CPU และ memory

การเก็บข้อมูล CPU และ memory ด้วย Grafana จะเริ่มจากให้ notebook ประมวลผลเสร็จ จากนั้นใน Grafana ให้เลือกช่วงเวลาที่ทำการประมวลผลของ notebook นั้น ๆ ซึ่งในการเลือกช่วงเวลาผู้วิจัยจะกำหนดเวลาของแต่ละไฟล์ไม่เกิน 300 วินาที หากไฟล์ไหนใช้ระยะเวลาประมวลผลนานเกิน 300 วินาที จะตัดข้อมูลที่เกินออก และบันทึกไฟล์ในรูปแบบ csv โดยในไฟล์จะทำการตัดข้อมูล container ที่ไม่จำเป็นออกดังตารางที่ 2 และ 3

ตาราง 2 ตัวอย่างไฟล์ csv ของ CPU ของ container

Time	jupyter-pariwat-pr	jupyter-zen-pariwat	jupyter-zen-pr	jupyter-zenji-pariwat	jupyter-zooksonline
2017-09-05T07:20:00.000Z	0.005454545	0.003454545	0.004909091	0.004	0.005454545
2017-09-05T07:20:10.000Z	0.019090909	0.091090909	0.113272727	0.041454545	0.065090909
2017-09-05T07:20:20.000Z	0.140181818	0.351090909	0.328	0.135272727	0.148181818
2017-09-05T07:20:30.000Z	0.209636364	0.841818182	0.829272727	0.425272727	0.726363636
2017-09-05T07:20:40.000Z	0.251454545	1.444545455	1.432	0.911454545	1.353454545
2017-09-05T07:20:50.000Z	0.569454545	1.898181818	1.96	1.609090909	1.879818182
2017-09-05T07:21:00.000Z	1.248909091	2.274909091	2.488727273	1.921090909	2.322181818
2017-09-05T07:21:10.000Z	1.832727273	2.739636364	2.840363636	2.355272727	2.774181818
2017-09-05T07:21:20.000Z	2.298	2.709818182	2.751272727	2.814545455	2.888545455
2017-09-05T07:21:30.000Z	2.726347175	2.627445277	2.462911788	2.772525634	2.641626064
2017-09-05T07:21:40.000Z	2.966727273	2.244181818	2.165454545	2.391090909	2.145090909
2017-09-05T07:21:50.000Z	2.755636364	2.16	1.901272727	2.076	2.032545455
2017-09-05T07:22:00.000Z	2.268	1.800363636	1.46	1.943272727	1.719272727
2017-09-05T07:22:10.000Z	2.086909091	1.432727273	1.356363636	1.697454545	1.388727273

ตาราง 3 ตัวอย่างไฟล์ csv ของ memory ของ container

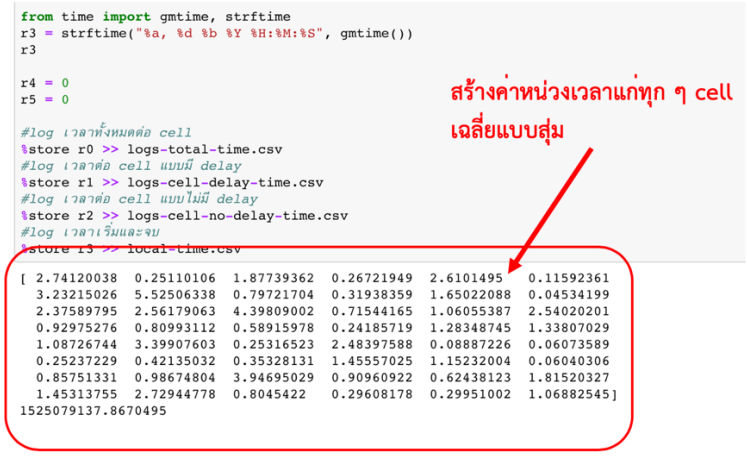
Time	jupyter-pariwat-pr	jupyter-zen-pariwat	jupyter-zen-pr	jupyter-zenji-pariwat	jupyter-zooksonline
2017-09-05T07:20:00.000Z	174362624	687255552	688123904	686272512	718237696
2017-09-05T07:20:10.000Z	295428096	1057873920	1157324800	860684288	998907904
2017-09-05T07:20:20.000Z	933130240	1635774464	1627582464	1163042816	1209454592
2017-09-05T07:20:30.000Z	1093656576	2428334080	2406584320	1765134336	2204028928
2017-09-05T07:20:40.000Z	1207128064	2881839104	2873499648	2535845888	2864574464
2017-09-05T07:20:50.000Z	1595457536	2994868224	3409252352	3101704192	3447775232
2017-09-05T07:21:00.000Z	2448556032	3460096000	3794071552	3151421440	4242223104
2017-09-05T07:21:10.000Z	2832465920	3956056064	4282380288	4322066432	4415528960
2017-09-05T07:21:20.000Z	3570290688	4584730624	4800106496	4977086464	5132095488
2017-09-05T07:21:30.000Z	4250955776	4823760896	4897333248	5387214848	5586665472
2017-09-05T07:21:40.000Z	4891295744	5101223936	4943441920	5871464448	5572943872
2017-09-05T07:21:50.000Z	5094830080	5393870848	4961140736	5934792704	6283677696
2017-09-05T07:22:00.000Z	5505372160	5491650560	5224140800	6283919360	6299504640
2017-09-05T07:22:10.000Z	5819084800	5499883520	5231202304	6369882112	6460592128

ทั้งตาราง 2 และ 3 นั้นจะแบ่งรายละเอียดออกเป็นสองส่วนคือส่วนของเวลาที่แสดง ปี-เดือน-วัน และเวลาที่แสดงชั่วโมง-นาที-วินาที ตามลำดับ และอีกส่วนคือ resource ที่ถูกใช้งาน โดยตาราง 2 เป็น resource ของ CPU ที่ถูกใช้งานในแต่ละ container มีหน่วยเป็น core และตาราง 3 เป็น resource ของ memory ที่ถูกใช้งานในแต่ละ container มีหน่วยเป็น bytes

3.3.3 การเก็บข้อมูลระยะเวลาที่ใช้ในการประมวลผลของ notebook ด้วย shell script

เนื่องจาก Prometheus สามารถเก็บระยะเวลาได้แต่หากไม่แม่นยำเนื่องจาก Prometheus จะเก็บทุก 5 วินาที ทางผู้วิจัยจึงเขียน code เฉพาะเพิ่มลงไปในไฟล์ notebook เพื่อเก็บเวลาให้การประมวลผลทุก cell และผู้วิจัยต้องการเห็นผลต่างหากไฟล์ notebook เดียวกันแต่เวลาต่อ cell ไม่เท่ากันแล้วการรองรับจำนวน notebook จะเพิ่มขึ้นหรือไม่ ดังนั้นใน code ที่ใส่เพิ่มในไฟล์ notebook จะมีดังนี้

1. หน่วงเวลาตอนเริ่ม – เพื่อให้ notebook ทุกตัวรันพร้อมกัน โดยให้รันพร้อมกันเมื่อนาที %5 = 0
2. จำลองค่าหน่วงเวลาออกมาให้ครบทุก cell ดังภาพประกอบ 22
3. สร้าง cell ขึ้นมาระหว่าง cell 2 cell และนำค่าหน่วงเวลานั้นมาใส่เพื่อหน่วงเวลาระหว่าง cell 2 cell ดังภาพประกอบ 23
4. บันทึกเวลาตั้งแต่เริ่มจนจบ
5. บันทึกเวลาการรันในแต่ละ cell โดยไม่รวมดีเลย์ในแต่ละ cell - โดยดีเลย์ระหว่าง cell ในแต่ละ notebook ผู้วิจัยจะแบ่งออกเป็น 1 วินาที, 5 วินาที และ 10 วินาที



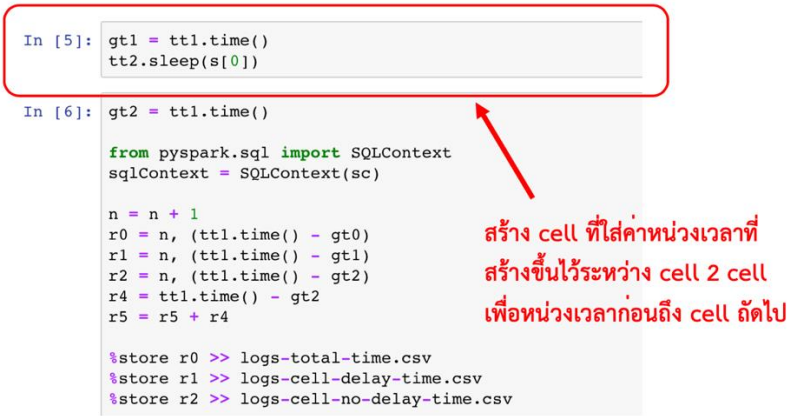

```
from time import gmtime, strftime
r3 = strftime("%a, %d %b %Y %H:%M:%S", gmtime())
r3

r4 = 0
r5 = 0

#log เวลาทั้งหมดต่อ cell
%store r0 >> logs-total-time.csv
#log เวลาต่อ cell แบบมี delay
%store r1 >> logs-cell-delay-time.csv
#log เวลาต่อ cell แบบไม่มี delay
%store r2 >> logs-cell-no-delay-time.csv
#log เวลาเริ่มและจบ
%store r3 >> local-time.csv
```

```
[ 2.74120038  0.25110106  1.87739362  0.26721949  2.6101495   0.11592361
  3.23215026  5.52506338  0.79721704  0.31938359  1.65022088  0.04534199
  2.37589795  2.56179063  4.39809002  0.71544165  1.06055387  2.54020201
  0.92975276  0.80993112  0.58915978  0.24185719  1.28348745  1.33807029
  1.08726744  3.39907603  0.25316523  2.48397588  0.08887226  0.06073589
  0.25237229  0.42135032  0.35328131  1.45557025  1.15232004  0.06040306
  0.85751331  0.98674804  3.94695029  0.90960922  0.62438123  1.81520327
  1.45313755  2.72944778  0.8045422   0.29608178  0.29951002  1.06882545]
1525079137.8670495
```

ภาพประกอบ 22 การสร้างค่า idle time สำหรับใช้หน่วงเวลา




```
In [5]: gt1 = tt1.time()
        tt2.sleep(s[0])

In [6]: gt2 = tt1.time()

        from pyspark.sql import SQLContext
        sqlContext = SQLContext(sc)

        n = n + 1
        r0 = n, (tt1.time() - gt0)
        r1 = n, (tt1.time() - gt1)
        r2 = n, (tt1.time() - gt2)
        r4 = tt1.time() - gt2
        r5 = r5 + r4

        %store r0 >> logs-total-time.csv
        %store r1 >> logs-cell-delay-time.csv
        %store r2 >> logs-cell-no-delay-time.csv
```

ภาพประกอบ 23 การหน่วงเวลาระหว่าง cell 2 cell

สำหรับวิธีการรัน notebook พร้อมกัน ผู้วิจัยใช้คำสั่งใน Application Terminal เพื่อให้รัน notebook ได้พร้อมกัน โดยคำสั่งที่ใช้คือ

```
docker exec –it names notebook jupyter nbconvert --to notebook --ExecutePreprocessor.enabled=True --ExecutePreprocessor.timeout=300 --execute path
```

เมื่อรันเสร็จแล้วผู้วิจัยจะใช้ shell script ที่เขียนไว้รันเพื่อเก็บไฟล์ที่ได้ในแต่ละ notebook มาเก็บไว้ในเครื่องเพื่อใช้ในการทำตัวแปรต่อไป

3.4 ตัวแปรที่ใช้สำหรับ Machine Learning Model

หลังจากเก็บข้อมูลเสร็จ จึงนำข้อมูลที่ได้มาทำเป็นตัวแปรเพื่อใช้ในการประมวลผลใน Machine Learning ตัวแปรที่ใช้จะมี 2 ส่วนหลักคือ ตัวแปรที่จำลองในส่วนผู้ใช้งาน(ข้อ 1, 2) และส่วนของการประมวลผลของ notebook (ข้อ 3, 4) โดยมีรายละเอียดดังนี้

1. number of current users (จำนวนผู้ใช้งาน ณ เวลานั้น) - จำนวนผู้ใช้เป็นตัวแปรสำคัญที่ใช้หาค่า response time เพราะถ้าจำนวนผู้ใช้มาก การประมวลผลการใช้งานของ notebook จะมากตาม และ server เองจำเป็นต้องเพิ่มประมวลผลด้านการจัดสรรทรัพยากรให้แต่ละ notebook ด้วย

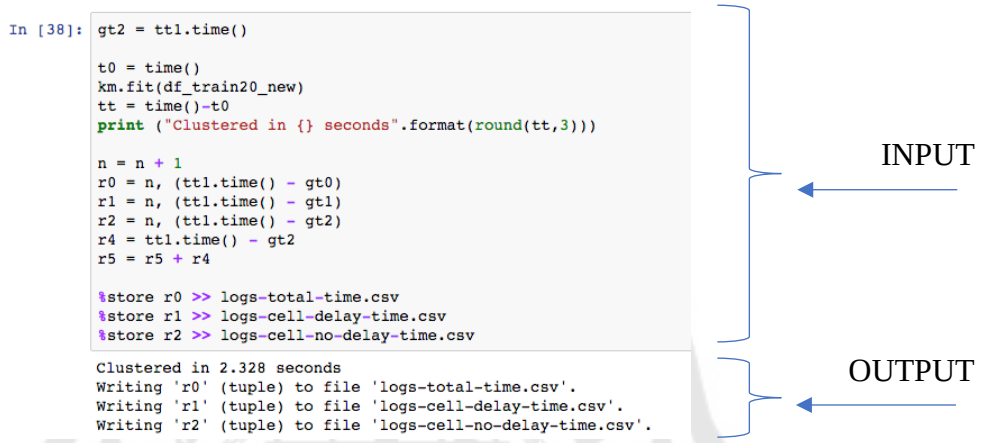
2. idle-time (เวลาเฉลี่ยระหว่าง cell) - เนื่องจากเวลาในการประมวลผลจะเริ่มเมื่อผู้ใช้งานสั่งให้ cell นั้นทำงาน จะมีช่วงเวลาก่อนหน้านั้นที่เครื่อง server จะยังไม่เริ่มประมวลผล (Input) ดังภาพประกอบ 24 ซึ่งเวลาในช่วงนี้ของผู้ใช้แต่ละคนจะไม่เท่ากัน และเนื่องจากผู้วิจัยใช้วิธีจำลองสถานการณ์การใช้สร้างของผู้ใช้ ทำให้ในช่วงเวลานี้จึงไม่มีค่า (เวลาที่ใช้ 0 วินาที) จึงต้องใช้ค่าอื่นจำลองเวลาช่วงนี้ขึ้นมา ซึ่งผู้วิจัยเลือกใช้การหน่วงเวลาระหว่าง cell (idle time) แทน โดยการหน่วงเวลานี้จะอยู่ในช่วง หลังจากที่เครื่องประมวลผล cell หนึ่งเสร็จ จะมีการหน่วงเวลาค่าหนึ่งมาหน่วงเวลาก่อนประมวลผล cell ถัดไป ซึ่งผู้วิจัยได้แบ่งการหน่วงเวลาออกเป็น 3 แบบ ดังตาราง 4

ตาราง 4 รายละเอียด Idle time

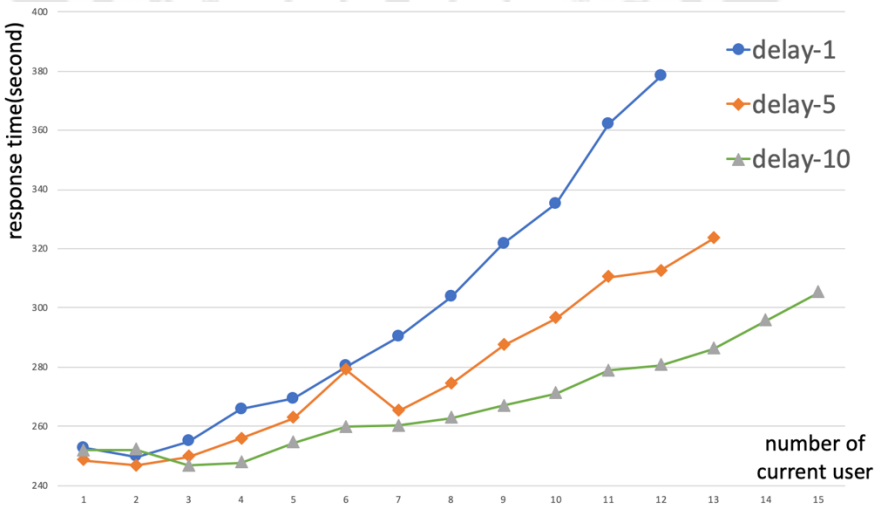
Idle time (วินาที)	รายละเอียด
1	แทนกรณีกลุ่มผู้ใช้งานส่วนใหญ่ input ข้อมูลเวลาใกล้เคียงกัน
5	ดีเลย์เฉลี่ย 5 วินาที แทนกรณีกลุ่มผู้ใช้งานส่วนใหญ่ input ข้อมูลเวลาต่างกัน
10	แทนกรณีกลุ่มผู้ใช้งานส่วนใหญ่ input ข้อมูลเวลาต่างกันมาก

สาเหตุที่เลือกใช้ตัวแปร Idle time อีกอย่างหนึ่งคือตัวแปรนี้มีผลกับการรองรับจำนวนผู้ใช้งาน ดังภาพประกอบ 25 จะสังเกตได้ว่าเมื่อค่า idle time เฉลี่ยเพิ่ม จำนวนการรองรับผู้ใช้ก็จะเพิ่มตาม สาเหตุที่ทำให้การรองรับผู้ใช้งานเพิ่มขึ้นเนื่องมาจากค่า idle time เฉลี่ยที่มากจะช่วยให้โอกาสการประมวลผลของ notebook พร้อมกันน้อยลง เช่น มี cell หนึ่งที่ใช้งาน CPU สูงถึง 6 core หากเครื่องมี CPU 24 core จะทำให้ช่วง cell นั้นสามารถรัน notebook ได้พร้อมกันแค่ 4

เครื่อง แต่หากมีค่า idle time ระหว่าง cell ของแต่ละ notebook ต่างกัน เมื่อ notebook ที่มีค่า idle time ระหว่าง cell น้อยสุดประมวลผลเสร็จจะคืนทรัพยากร CPU ที่ใช้ให้กับ server เพื่อให้ server นำทรัพยากรนี้ไปประมวลผล notebook อื่นต่อ ทำให้การรองรับการใช้งาน notebook เพิ่มขึ้น



ภาพประกอบ 24 รายละเอียดการแบ่งประเภทระหว่าง idle time และ response time



ภาพประกอบ 25 กราฟแสดงการใช้งานของไฟล์ notebook

3. AVG-memory-per-user (ค่าเฉลี่ยการใช้ memory ต่อผู้ใช้) - ค่าเฉลี่ย memory ที่ใช้ประมวลผลจะสัมพันธ์กับจำนวนผู้ใช้ เพราะยิ่งผู้ใช้มาก ค่าเฉลี่ย memory ที่ใช้ประมวลผลจะมากตามและส่งผลต่อ response time ที่มากขึ้นเช่นกัน สำหรับข้อมูล memory ที่ใช้ทั้งสามไฟล์

4. AVG-CPU-per-user (ค่าเฉลี่ยการใช้ CPU ต่อผู้ใช้) - ค่าเฉลี่ย CPU ที่ใช้ประมวลผลจะสัมพันธ์กับจำนวนผู้ใช้ เพราะยิ่งผู้ใช้มาก ค่าเฉลี่ย CPU ที่ใช้ประมวลผลจะมากตามและส่งผลต่อ response time ที่มากขึ้นเช่นกัน สำหรับข้อมูล CPU ที่ใช้ทั้งสามไฟล์

3.5 Hyper-parameter ที่ใช้สำหรับการทำนายข้อมูล

3.5.1 KNN Regression

ในส่วนการใช้ KNN Regression เพื่อทำนายระยะเวลาในการรัน notebook เบื้องต้นจะเซ็ตค่าพารามิเตอร์ KNN Regression ดังนี้

1. n_neighbors: 5, 10, 15, 20
2. leaf_size: 10, 30, 50, 70, 100
3. weights: uniform, distance
4. P: 1, 2

สาเหตุที่ใช้ n_neighbors และ leaf_size เท่านี้เนื่องจากเมื่อใส่ค่าที่มากกว่านี้จะเกิดการ overfitting ทำให้ค่าทำนายที่ได้ต่ำ

3.5.2 Random Forest Regression

ในส่วนการใช้ Random Forest Regression เพื่อทำนายระยะเวลาในการรัน notebook เบื้องต้นจะเซ็ตค่าพารามิเตอร์ Random Forest Regression ดังนี้

1. n_estimators: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
2. random_states: 1, 5, 10, 15, 20
3. max_features: auto, sqrt, log2
4. max_leaf_nodes: None, 5, 10, 15, 20, 25, 30, 50

สาเหตุที่ใช้ n_ estimators, random_states และ max_leaf_nodes เท่านี้เนื่องจากเมื่อใส่ค่าที่มากกว่านี้จะเกิดการ overfitting ทำให้ค่าทำนายที่ได้ต่ำ

บทที่ 4
ผลลัพธ์จากการวิจัย

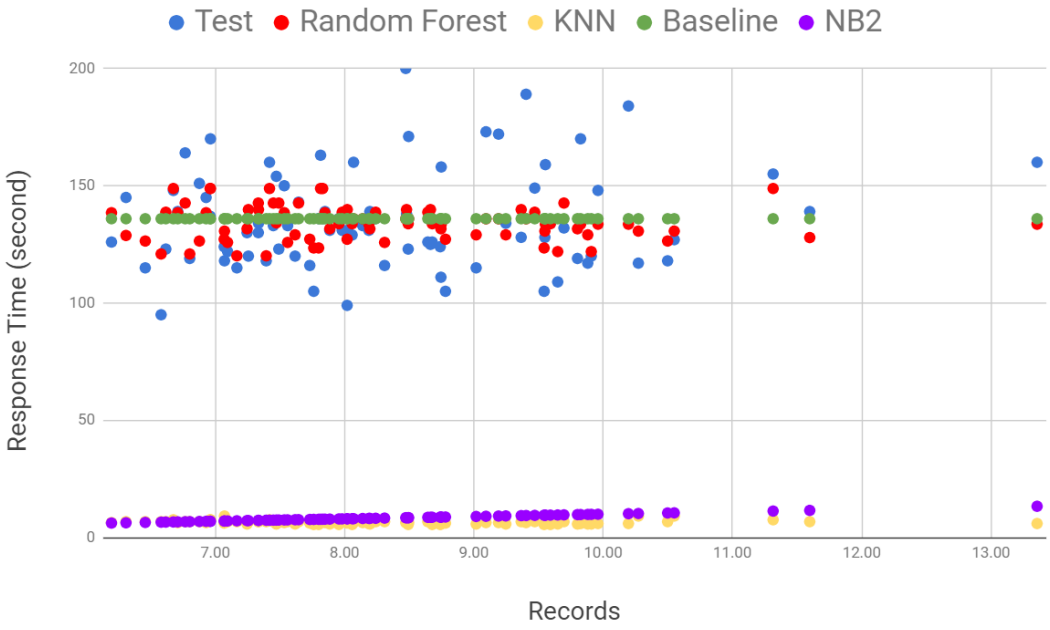
หลังจากทำนาย response time ด้วยเทคนิคการเรียนรู้ของเครื่อง ผู้วิจัยจึงเปรียบเทียบประสิทธิภาพเทคนิคการเรียนรู้ของเครื่องระหว่าง Random Forest Regression และ KNN Regression กับ Baseline ที่ใช้ค่า response time ที่ได้จากไฟล์ NBTest ของ user 1 คนมาเฉลี่ยเพื่อทำนายหาค่า response time ของ NBTest โดยทั้ง Random Forest Regression และ KNN Regression ได้ใช้ 5-fold cross-validate ในการจูนค่า hyper-parameter เพื่อนำไปใช้ใน machine learning model เพื่อให้ได้ผลลัพธ์การทำนายที่ดีที่สุดโดยค่าที่ได้จากการ tuning จะแสดงดังตาราง 5

ตาราง 5 Hyper-parameter ของแต่ละ machine learning model

Machine learning Model	Metric	Hyper-parameter									
Random Forest	n_estimators	1	2	5	6	**10**	12	20	25	30	50
	random_states	**1**	5	10	15	20					
	max_features	auto	**sqrt**	log2							
	max_leaf_nodes	**None**	5	10	15	20	50				
KNN	weights	uniform	**distance**								
	n_neighbors	**5**	10	15	20						
	p	1	**2**								
	leaf_size	**10**	30	50	70	100					

จากในตาราง 5 จะแสดงค่า Hyper-parameter ที่ดีที่สุดที่ได้จากการใช้ 5-fold cross-validate โดย Random Forest จะได้ Metric n_estimators = 10, random_states = 1, max_features = sqrt และ max_leaf_nodes = None ส่วน KNN จะได้ Metric weights = distance, n_neighbors = 5, P = 2 และ leaf_size = 10 สำหรับผลลัพธ์จากการ cross-validate

นั้น Random forest วัดความแม่นยำด้วย MAPE = 12.203% และ KNN วัดความแม่นยำด้วย MAPE = 11.231% เมื่อเราหา Hyper-parameter ได้แล้ว จึงนำ training data มาเข้ากระบวนการ machine learning ของ Random forest และ KNN จากนั้นจึงนำโมเดลที่ได้ มาทดสอบกับ testing data เพื่อหา response time ของ test data ซึ่งก็คือ NBTest โดยผลลัพธ์การทำนายนั้นออกมาดังตาราง 6 และตาราง 7 ซึ่งภายในตารางจะมี column ที่ประกอบไปด้วย column Test คือค่า response time จริงของ test data, column Random Forest คือค่าทำนาย response time ที่ได้จาก random forest, column KNN คือค่าทำนาย response time ที่ได้จาก KNN และ column Baseline คือค่า response time ที่ได้จากการนำค่า response time ของไฟล์ NB1 มาเฉลี่ย เมื่อนำผลลัพธ์ที่ได้มาสร้างเป็นกราฟ จะได้กราฟดังภาพประกอบ 26



ภาพประกอบ 26 กราฟแสดงผลลัพธ์จากการทำนายค่า Response time ของ NBTest โดยใช้ Random forest, KNN และ Baseline

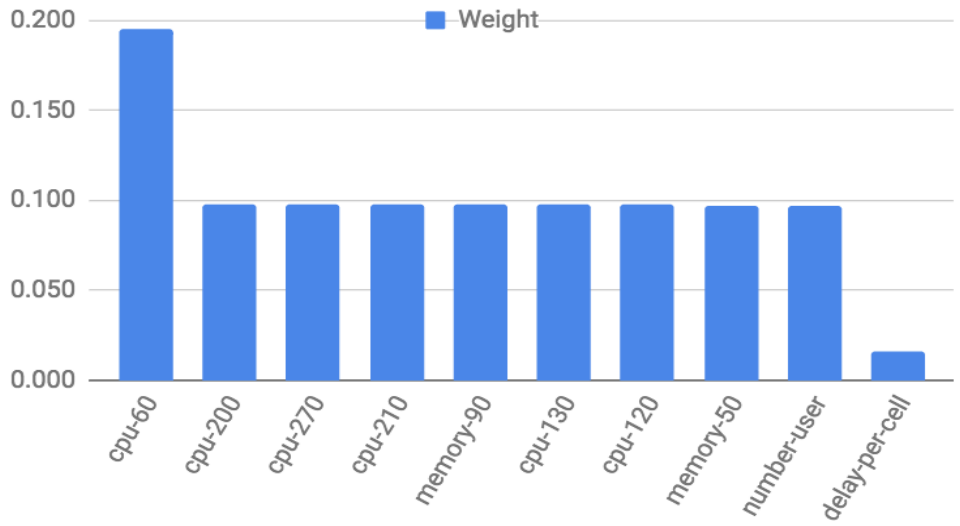
ตาราง 6 ผลลัพธ์การทำนาย response time ของแต่ละ model (วินาที) ส่วนที่ 1

No.	Test	Random Forest	KNN	Baseline
1	145.00	128.80	6.80	135.94
2	95.00	120.92	6.80	135.94
3	119.00	120.92	6.80	135.94
4	118.00	126.43	6.80	135.94
5	115.00	126.43	6.80	135.94
6	151.00	126.43	6.80	135.94
7	118.00	130.67	9.20	135.94
8	117.00	130.67	9.20	135.94
9	127.00	130.67	9.20	135.94
10	128.00	130.67	9.20	135.94
11	126.00	138.50	6.40	135.94
12	150.00	138.50	6.40	135.94
13	145.00	138.50	6.40	135.94
14	139.00	138.50	6.40	135.94
15	139.00	138.50	6.40	135.94
16	132.00	142.62	6.80	135.94
17	143.00	142.62	6.80	135.94
18	164.00	142.62	6.80	135.94
19	123.00	142.62	6.80	135.94
20	130.00	142.62	6.80	135.94
21	133.00	142.62	6.80	135.94
22	163.00	148.82	7.60	135.94
23	136.00	148.82	7.60	135.94
24	160.00	148.82	7.60	135.94
25	155.00	148.82	7.60	135.94
26	170.00	148.82	7.60	135.94
27	137.00	148.82	7.60	135.94
28	148.00	148.82	7.60	135.94
29	139.00	127.90	6.80	135.94
30	115.00	120.13	6.80	135.94
31	118.00	120.13	6.80	135.94
32	122.00	125.83	6.80	135.94
33	116.00	125.83	6.80	135.94
34	133.00	125.83	6.80	135.94
35	124.00	127.19	6.20	135.94
36	116.00	127.19	6.20	135.94
37	99.00	127.19	6.20	135.94
38	105.00	127.19	6.20	135.94
39	123.00	138.71	6.80	135.94
40	149.00	138.71	6.80	135.94
41	136.00	138.71	6.80	135.94
42	126.00	138.71	6.80	135.94

ตาราง 7 ผลลัพธ์การทำนาย response time ของแต่ละ model (วินาที) ส่วนที่ 2

No.	Test	Random Forest	KNN	Baseline
43	131.00	138.71	6.80	135.94
44	131.00	139.83	6.80	135.94
45	120.00	139.83	6.80	135.94
46	134.00	139.83	6.80	135.94
47	128.00	139.83	6.80	135.94
48	125.00	139.83	6.80	135.94
49	138.00	139.83	6.80	135.94
50	189.00	135.91	6.40	135.94
51	133.00	135.91	6.40	135.94
52	160.00	135.91	6.40	135.94
53	171.00	135.91	6.40	135.94
54	172.00	135.91	6.40	135.94
55	173.00	135.91	6.40	135.94
56	200.00	135.91	6.40	135.94
57	154.00	134.17	5.80	135.94
58	120.00	121.93	5.80	135.94
59	109.00	121.93	5.80	135.94
60	136.00	123.48	5.60	135.94
61	105.00	123.48	5.60	135.94
62	105.00	123.48	5.60	135.94
63	120.00	129.07	5.80	135.94
64	115.00	129.07	5.80	135.94
65	134.00	129.07	5.80	135.94
66	117.00	129.07	5.80	135.94
67	130.00	131.65	5.80	135.94
68	139.00	131.65	5.80	135.94
69	111.00	131.65	5.80	135.94
70	131.00	131.65	5.80	135.94
71	119.00	131.65	5.80	135.94
72	136.00	133.80	5.60	135.94
73	123.00	133.80	5.60	135.94
74	124.00	133.80	5.60	135.94
75	134.00	133.80	5.60	135.94
76	126.00	133.80	5.60	135.94
77	129.00	133.80	5.60	135.94
78	148.00	133.60	6.00	135.94
79	159.00	133.60	6.00	135.94
80	170.00	133.60	6.00	135.94
81	160.00	133.60	6.00	135.94
82	131.00	133.60	6.00	135.94
83	158.00	133.60	6.00	135.94
84	184.00	133.60	6.00	135.94

จากกราฟในภาพประกอบ 26 สีฟ้าคือ testing data (NBTest), สีแดงคือ Random Forest, สีเหลืองคือ KNN, สีเขียวคือ baseline และสีม่วงคือ NB2 ซึ่งจะเห็นได้ว่า Random Forest (สีแดง) ใกล้เคียง testing data มากที่สุด ในขณะที่ KNN (สีเหลือง) ใกล้เคียง testing data (NBTest) น้อยที่สุด สำหรับสาเหตุที่ Random Forest มีค่าดีที่สุด เนื่องจาก Random Forest มีการใช้ฟังก์ชั่นในการให้น้ำหนัก feature ที่มีผลในการแบ่งประเภทของ dataset ที่เรียกว่า Feature Importance



ภาพประกอบ 27 Feature Importance ของ Random Forest

จากภาพประกอบ 27 feature cpu-60 (ค่า CPU ในวินาทีที่ 60) มีการให้น้ำหนักสูงที่สุดที่มีค่าเท่ากับ 0.198 รองลงมาคือ cpu-200 cpu-210 cpu-270 memory-90 cpu-130 cpu-120 memory-50 number-user ที่มีค่าเท่ากับ 0.098 และสุดท้าย delay-per-cell ที่มีค่าเท่ากับ 0.023 ที่ให้น้ำหนักน้อยสุด ต่างจาก KNN ที่ใช้วิธี nearest neighbor หากลุ่มที่ใกล้ที่สุด ดังตาราง 8

ตาราง 8 ระยะห่างเฉลี่ยของ NB1, NB2, NB3 กับ NBTest

Training set	ระยะห่างเฉลี่ยระหว่าง NBTest
NB1	19.742
NB2	15.942
NB3	45.860

จากตาราง 8 จะเห็นว่าค่าที่น้อยที่สุดคือ NB2 เนื่องจากหากดูในภาพประกอบ 21 จะเห็นว่าค่า cpu ของ NBTest ในช่วง 170 วินาทีเป็นต้นไปมีค่าเป็น 0 เพราะการรันสิ้นแล้ว ซึ่งใกล้กับค่า cpu ของ NB2 ที่เข้าใกล้ 0 เช่นกัน และหากดูในภาพประกอบ 20 แล้ว ค่า memory ของ NBTest ก็ใกล้กับค่า memory ของ NB2 เช่นกัน จึงทำให้ KNN ให้น้ำหนักของ NB2 ค่อนข้างมาก และทำนายค่า response time ใกล้เคียงกับ NB2 ดังกราฟในภาพประกอบ 26 (สีม่วง) คือค่า response time ของ NB2 และสีเหลืองคือ response time ของ KNN ซึ่งมีค่าใกล้เคียงกับ NB2 และเมื่อใช้ MAE และ MAPE ในการวัดความแม่นยำดังตาราง 9 Random Forest จะได้ค่าความแม่นยำทั้ง MAE และ MAPE ดีที่สุด โดย MAE ได้ 13.679 second และ MAPE ได้ค่าความแม่นยำ 9.732 % ในขณะที่ KNN นั้นได้ค่าความแม่นยำต่ำที่สุด โดย MAE ได้ 128.907 second และ MAPE ได้ 95.057 % ส่วน Baseline ได้ค่าความแม่นยำ MAE 16.145 second และ MAPE ได้ 11.935 % สาเหตุที่ Baseline มีความแม่นยำได้ดีกว่า KNN เนื่องจาก KNN นั้นทำนายค่า response time ผิด (ทำนายใกล้เคียงกับ NB2 แทน) และ Baseline ใช้ค่า response time เฉลี่ยของ NBTest ที่ทำงานเมื่อมีผู้ใช้ 1 คนแต่ค่าที่ได้จากการทำนายนั้นมีแค่ค่าเดียว จึงได้ค่าความแม่นยำน้อยกว่า Random Forest ที่ค่าที่ได้จากการทำนายนั้นจะขึ้นลงตามจำนวนผู้ใช้

ส่วนค่าความแม่นยำที่ได้จากการ cross-validate ใน Random Forest นั้นได้ 12.203 % ซึ่งแย่กว่าค่า MAPE Test data ที่ได้ 9. 732 % เนื่องจากการทำ cross-validate นั้นจะแบ่ง Train data ออกเป็นกลุ่มย่อย ๆ (ในงานวิจัยนี้แบ่งออกเป็น 5 กลุ่ม) แล้วหาค่าการทำนายที่ดีที่สุด แต่การแบ่งกลุ่มนั้นทำให้ข้อมูลในแต่ละ NB (NB1, NB2, NB3) กระจายออกไปในแต่ละกลุ่มด้วย จึงทำให้ข้อมูลที่อยู่ในกลุ่มไม่ไปในทิศทางเดียวกัน ต่างจากการทำนาย Test data ที่นำ NBTest มาเป็น Test data เพียงอย่างเดียว จึงอยู่ในรูปแบบเดียวกัน แบ่งแยกประเภทของ NB ได้ง่ายกว่า ทำให้การทำนายมีโอกาสถูกต้องมากกว่าการทำนายใน cross-validate

ตาราง 9 วัดความแม่นยำด้วย MAE และ MAPE

Performance index	Description	Prediction model		
		Random Forest	KNN	Baseline
MAPE (%)	Cross-validate	12.203	11.231	N/A
	Test dataset	9.732	95.057	11.935
MAE (second)	Test dataset	13.679	128.907	16.145

บทที่ 5
สรุปผลการวิจัย อภิปรายผล และข้อเสนอแนะ

5.1 สรุปผลการวิจัย

ในงานวิจัยนี้นำเสนอการใช้ Machine Learning Model ในการทำนายประสิทธิภาพการใช้งาน Jupyter Notebook ที่ทำงานอยู่บน JupyterHub โดย model ที่ใช้คือ supervised learning (KNN, Random Forest) เปรียบเทียบกับ Baseline Model ซึ่งเป็นการนำค่าเฉลี่ยของข้อมูลผู้ใช้หนึ่งคนมาทำนาย โดย Machine Learning Model และ Baseline Model จะทำนาย response time ของ Jupyter notebook และผลลัพธ์ที่ได้คือ Random Forest model ทำนายได้ดีที่สุด โดยได้ค่า MAE เท่ากับ 13.679 second และค่า MAPE เท่ากับ 9.732% รองลงมาคือ Baseline โดยได้ค่า MAE เท่ากับ 16.145 second และค่า MAPE เท่ากับ 11.935% และ model ที่ทำนายแย่ที่สุดคือ KNN โดยได้ค่า MAE เท่ากับ 128.907 second และค่า MAPE เท่ากับ 95.057%

5.2 อภิปรายผล

ผลลัพธ์จากการวิจัยการใช้ Machine Learning Model ในการทำนายประสิทธิภาพการใช้งาน Jupyter Notebook ที่ทำงานอยู่บน JupyterHub มีประเด็นที่จะนำมาเสนอ ดังนี้

1. สามารถนำ Random forest มาใช้ในการทำนายประสิทธิภาพการใช้งาน Jupyter Notebook ที่ทำงานอยู่บน JupyterHub ได้ เนื่องจาก Random forest มีค่าผิดพลาดในการทำนายอยู่ที่ 9.732% ได้เมื่อวัดด้วย MAPE และมีค่าผิดพลาดในการทำนาย 13.679 second หากวัดด้วย MAE โดยสิบอันดับ feature ที่สำคัญที่จะช่วยให้ random forest ทำนายได้ดีคือ cpu-60, cpu-220, cpu-270, cpu-210, cpu-90, memory-90, cpu-130, cpu-120, memory-50, number-user และ delay-per-cell ตามลำดับ

2. KNN ไม่สามารถนำมาใช้ในการทำนายประสิทธิภาพการใช้งาน Jupyter Notebook ที่ทำงานอยู่บน JupyterHub ได้ เนื่องจาก KNN มีค่าผิดพลาดในการทำนายอยู่ที่ 95.057% ได้เมื่อวัดด้วย MAPE และมีค่าผิดพลาดในการทำนาย 128.907 second สาเหตุที่ผิดพลาดเนื่องจาก KNN แยกประเภทของ NB ไม่ถูก เพราะค่า cpu และ memory ของ NBTest ใกล้กับ NB2 ทำให้แยกประเภทของ NB ผิด เมื่อแยกประเภทผิด ทำให้การทำนายผิดตามไปด้วย

5.3 ข้อเสนอแนะ

สำหรับข้อเสนอแนะเพื่อนำงานวิจัยการทำนายประสิทธิภาพการใช้งาน Jupyter Notebook ที่ทำงานอยู่บน JupyterHub นี้ไปใช้ในการศึกษาค้นคว้าต่อไปมีดังนี้

1. ในงานวิจัยเก็บและใช้ข้อมูลสำหรับทำเทคนิคการเรียนรู้ของเครื่องจำนวน 993 records และใช้ NB ไฟล์จำนวน 4 ไฟล์แบ่งเป็น 3 ไฟล์สำหรับ train และ 1 ไฟล์สำหรับ test หากต้องการให้การทำนายแม่นยำและถูกต้องที่ดีและน่าเชื่อถือมากขึ้น จำเป็นต้องเก็บข้อมูลมากกว่านี้

2. ในการวัดความแม่นยำของเทคนิคการเรียนรู้ของเครื่องในงานวิจัยนี้ใช้ response time เป็นตัวชี้วัดประสิทธิภาพการทำงานด้านระยะเวลาในการประมวลผลของ Jupyter Notebook ที่ทำงานอยู่บน JupyterHub หากต้องการเพิ่มประสิทธิภาพในการทำนายและสามารถใช้ในการเรียนการสอนได้จริง จำเป็นต้องมีตัวชี้วัดความเสี่ยงของจำนวนผู้ใช้งานด้วย เนื่องจากหากเราสามารถทำนายได้ว่า NB ไฟล์ที่นำมาใช้รองรับผู้ใช้งานได้สูงสุดกี่คน ก็จะสามารถให้คำตอบในด้านประสิทธิภาพการรองรับผู้ใช้งานได้ เพื่อนำผลลัพธ์ที่ได้จากการทำนายนี้ไปใช้ในการทำนายการรองรับผู้เรียนได้จริง

3. การเก็บข้อมูลในวิจัยนี้เป็นการเก็บข้อมูลจากการจำลองสถานการณ์โดยใช้ idle time ซึ่งมีการกระจายตัวแบบ exponentially distributed เป็นตัวแทนระยะเวลาการใช้งานของผู้ใช้จำนวน 3 กลุ่ม ในอนาคตควรมีการเก็บข้อมูลจากผู้ใช้จริงเพื่อยืนยันสมมุติฐานนี้ต่อไป

บรรณานุกรม

1. Nagar, S., *Introduction to Python for Engineers and Scientists.* 2018.
2. Hamrick, J.B. *Creating and Grading IPython/Jupyter Notebook Assignments with NbGrader*. in *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. 2016. ACM.
3. Randles, B.M., et al. *Using the Jupyter Notebook as a tool for open science: An empirical study*. in *Digital Libraries (JCDL), 2017 ACM/IEEE Joint Conference on*. 2017. IEEE.
4. Ragan-Kelley, M., et al. *The Jupyter/IPython architecture: a unified view of computational research, from interactive exploration to communication and publication*. in *AGU Fall Meeting Abstracts*. 2014.
5. Seo, K.-T., et al., *Performance comparison analysis of linux container and virtual machine for building cloud.* Advanced Science and Technology Letters, 2014. **66**(105-111): p. 2.
6. Preeth, E., et al. *Evaluation of Docker containers based on hardware utilization*. in *Control Communication & Computing India (ICCC), 2015 International Conference on*. 2015. IEEE.
7. Vohra, D., *Kubernetes microservices with Docker*. 2016: Apress.
8. Betke, E. and J. Kunkel. *Real-time I/O-monitoring of HPC applications with SIOX, elasticsearch, Grafana and FUSE*. in *International Conference on High Performance Computing*. 2017. Springer.
9. Silver, D., et al., *Mastering the game of Go without human knowledge.* Nature, 2017. **550**(7676): p. 354.
10. Maia, J., et al., *Performance Comparison between Edited kNN and MQ-RBFN for Regression and Classification Tasks.*
11. Smith, P.F., S. Ganesh, and P. Liu, *A comparison of random forest regression and multiple linear regression for prediction in neuroscience.* Journal of neuroscience methods, 2013. **220**(1): p. 85-91.
12. Couceiro, M., P. Romano, and L. Rodrigues. *A Machine Learning Approach to*

Performance Prediction of Total Order Broadcast Protocols. in *2010 Fourth IEEE International Conference on Self-Adaptive and Self-Organizing Systems*. 2010.

13. Didona, D., et al. *Enhancing performance prediction robustness by combining analytical modeling and machine learning*. in *Proceedings of the 6th acm/spec international conference on performance engineering*. 2015. ACM.
14. McNaughton, M.J., *Predictive Pod Auto-scaling in the Kubernetes Container Cluster Manager.* 2016.
15. Sudalaikkan, L.V., *Preservation of low latency service request processing in dockerized microservice architectures*. 2016, Colorado State University. Libraries.

ภาคผนวก

รายละเอียดของ NB1, NB2, NB3 และ NBTest สามารถเข้าไปดูได้ที่ URL https://drive.google.com/drive/folders/1yDSlZ5FsunAbOMpFu_838XsdvqONiO2I?usp=sharing โดยภายในจะมีไฟล์ 2 ไฟล์ 1 คือไฟล์ที่มีนามสกุล zip ภายในจะมีไฟล์ NB1, NB2, NB3 และ NBTest ที่ใช้สำหรับงานวิจัย และไฟล์ที่มีนามสกุล csv คือรายละเอียดของข้อมูลทั้งหมดซึ่งภายในจะมีคอลัมน์ต่าง ๆ ดังนี้

1. no. – อันดับที่บันทึกข้อมูล
2. notebook – ประเภทของไฟล์ 1 คือ NB1, 2 คือ NB2, 3 คือ NB3 และ4 คือ NBTest
3. number-user – จำนวน user ที่ใช้งาน Jupyter notebook container ในขณะนั้น
4. delay-per-cell – idel time เฉลี่ยที่ใช้หน่วงเวลา
5. response-time – ระยะเวลาในการประมวลผลทั้งหมดของ Jupyter notebook container ในขณะนั้น
6. avg-cpu-per-user – resource CPU เฉลี่ยที่ถูกใช้ขณะนั้น
7. cpu-use(max) – resource CPU สูงสุดที่ถูกใช้ขณะนั้น
8. avg-memory-per-user – resource memory เฉลี่ยที่ถูกใช้ขณะนั้น
9. memory-use(max) – resource memory สูงสุดที่ถูกใช้ขณะนั้น
10. server-risk – ระดับความเสี่ยงของ server มีสองระดับคือ normal คือปกติและ risk คือ เสี่ยง server ล่ม โดย normal จะคิดเป็นการใช้ resource น้อยกว่าหรือเท่ากับ 80% และ risk คิดเป็นการใช้ resource มากกว่า 80%
11. cpu-10 ถึง cpu-300 – ข้อมูลการเก็บการใช้งาน CPU ทุก 10 วินาที มีทั้งหมด 30 คอลัมน์
12. memory-10 ถึง memory-300 – ข้อมูลการเก็บการใช้งาน memory ทุก 10 วินาที มีทั้งหมด 30 คอลัมน์

ตาราง 10 ตัวอย่างรายละเอียดของ NB1, NB2, NB3 และ NBTest

no.	notebook	number-user	delay-per-cell	response-time	AVG-cpu-per-user	cpu-user(max)	AVG-memory-per-user (GB)	memory-use-max (GB)	server-risk	cpu-10...300	memory-10...300
1	1	1	1	252.65	1.61	3.52	5.89	11.66	normal	0...3.52	0.09...11.66
2	1	2	1	250.49	1.61	3.52	5.89	11.66	normal	0...3.52	0.09...11.66
3	1	2	1	248.99	1.61	3.52	5.89	11.66	normal	0...3.52	0.09...11.66
4	1	3	1	244.13	1.61	3.52	5.89	11.66	normal	0...3.52	0.09...11.66
5	1	3	1	272.95	1.61	3.52	5.89	11.66	normal	0...3.52	0.09...11.66
6	1	3	1	247.67	1.61	3.52	5.89	11.66	normal	0...3.52	0.09...11.66
7	1	4	1	262.88	1.61	3.52	5.89	11.66	normal	0...3.52	0.09...11.66
8	1	4	1	262.35	1.61	3.52	5.89	11.66	normal	0...3.52	0.09...11.66
9	1	4	1	282.94	1.61	3.52	5.89	11.66	normal	0...3.52	0.09...11.66
10	1	4	1	254.85	1.61	3.52	5.89	11.66	normal	0...3.52	0.09...11.66
11	1	5	1	285.43	1.61	3.52	5.89	11.66	normal	0...3.52	0.09...11.66
12	1	5	1	266.98	1.61	3.52	5.89	11.66	normal	0...3.52	0.09...11.66
.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.
993	4	7	10	184.76	3.36	4.67	0.77	0.82	risk	0...3.99	0.19...0.76

ตัวอย่างของ NB1, NB2, NB3 และ NBTest

```
In [28]: gt2 = tt1.time()
         # 'flag' nominal column line 12
         print(train_df.select(nominal_cols[2]).distinct().count())
         (train_df.crosstab(nominal_cols[2], 'labels2').sort(sql.asc(nominal_cols[2] + '_labels2')).show())
         (train_df.crosstab(nominal_cols[2], 'labels5').sort(sql.asc(nominal_cols[2] + '_labels5')).show())

         n = n + 1
         r0 = n, (tt1.time() - gt0)
         r1 = n, (tt1.time() - gt1)
         r2 = n, (tt1.time() - gt2)
         r4 = tt1.time() - gt2
         r5 = r5 + r4

         %store r0 >> logs-total-time.csv
         %store r1 >> logs-cell-delay-time.csv
         %store r2 >> logs-cell-no-delay-time.csv
```

```
11
+------------+------+------+
|flag_labels2|attack|normal|
+------------+------+------+
|         OTH|     4|     1|
|         REJ|  1701|   515|
|        RSTO|   260|    44|
|      RSTOS0|    21|     0|
|        RSTR|   469|    28|
|          S0|  6929|    80|
|          S1|     2|    86|
|          S2|     4|    17|
|          S3|     0|    15|
|          SF|  2310| 12663|
|          SH|    43|     0|
+------------+------+------+
```

ภาพประกอบ 28 ตัวอย่าง NB1 (ส่วนที่ 1)

```
In [48]: gt2 = tt1.time()
         #line 25
         def getAttributeRatio(df, numericCols, binaryCols, labelCol):
             ratio_dict = {}

             if numericCols:
                 avg_dict = (df
                         .select(list(map(lambda c: sql.avg(c).alias(c), numericCols)))
                         .first()
                         .asDict())

                 ratio_dict.update(df
                         .groupBy(labelCol)
                         .avg(*numericCols)
                         .select(list(map(lambda c: sql.max(col('avg(' + c + ')')/avg_dict[c]).alias(c), numericCols)))
                         .fillna(0.0)
                         .first()
                         .asDict())

             if binaryCols:
                 ratio_dict.update((df
                         .groupBy(labelCol)
                         .agg(*list(map(lambda c: (sql.sum(col(c))/(sql.count(col(c)) - sql.sum(col(c)))).alias(c), binaryCols))
                         .fillna(1000.0)
                         .select(*list(map(lambda c: sql.max(col(c)).alias(c), binaryCols)))
                         .first()
                         .asDict()))

             return OrderedDict(sorted(ratio_dict.items(), key=lambda v: -v[1]))
```

ภาพประกอบ 29 ตัวอย่าง NB1 (ส่วนที่ 2)

```
gt2 = tt1.time()

Numberic = [
    "duration","src_bytes",
    "dst_bytes","wrong_fragment","urgent","hot","num_failed_logins",
    "num_compromised","num_root",
    "num_file_creations","num_shells","num_access_files","num_outbound_cmds",
    "count","srv_count","serror_rate",
    "srv_serror_rate","rerror_rate","srv_rerror_rate","same_srv_rate",
    "diff_srv_rate","srv_diff_host_rate","dst_host_count","dst_host_srv_count",
    "dst_host_same_srv_rate","dst_host_diff_srv_rate","dst_host_same_src_port_rate",
    "dst_host_srv_diff_host_rate","dst_host_serror_rate","dst_host_srv_serror_rate",
    "dst_host_rerror_rate","dst_host_srv_rerror_rate"
]

num_features = [
    "duration","src_bytes",
    "dst_bytes","land","wrong_fragment","urgent","hot","num_failed_logins",
    "logged_in","num_compromised","root_shell","su_attempted","num_root",
    "num_file_creations","num_shells","num_access_files",
    "is_host_login","is_guest_login","count","srv_count","serror_rate",
    "srv_serror_rate","rerror_rate","srv_rerror_rate","same_srv_rate",
    "diff_srv_rate","srv_diff_host_rate","dst_host_count","dst_host_srv_count",
    "dst_host_same_srv_rate","dst_host_diff_srv_rate","dst_host_same_src_port_rate",
    "dst_host_srv_diff_host_rate","dst_host_serror_rate","dst_host_srv_serror_rate",
    "dst_host_rerror_rate","dst_host_srv_rerror_rate","label2-index"
]

features = [
    "duration","src_bytes",
    "dst_bytes","land","wrong_fragment","urgent","hot","num_failed_logins",
    "logged_in","num_compromised","root_shell","su_attempted","num_root",
    "num_file_creations","num_shells","num_access_files",
```

ภาพประกอบ 30 ตัวอย่าง NB2 (ส่วนที่ 1)

```
In [ ]: gt2 = tt1.time()

for i in range(k):

    print ("Cluster {} labels:".format(i))
    print (label_names[i].value_counts())

       #Merge dataframe
    df_tran20 = pd.concat([df_train_20_features, label_names[i]],axis=1, join_axes=[label_names[i].index])
    df_tran20.columns.values[38] = 'cluster'
    y_df_train20 = df_tran20['label2-index']


    from sklearn.model_selection import train_test_split
    X_train, X_test, y_train, y_test = train_test_split(df_tran20, y_df_train20, test_size=0.30, random_state=0)

    X_train = X_train[features]
    X_test = X_test[features]

    # Random Forests Classifer
    from sklearn.metrics import accuracy_score
    from sklearn.ensemble import RandomForestClassifier
    from sklearn.metrics import classification_report
    rf = RandomForestClassifier( max_depth=None, n_estimators=10, min_samples_split=2)
    rf = rf.fit(X_train, y_train)
    trainPredict = rf.predict(X_train)
    print ("Random Forests Classifer")
    print("Accuracy Train: %.2f%%" % (accuracy_score(trainPredict, y_train)*100.0))
    pred = rf.predict(X_test)
    print("Accuracy Test: %.2f%%" % (accuracy_score(pred, y_test)*100.0))
    print('-------------')
    print(classification_report(y_test, pred, digits = 3))
```

ภาพประกอบ 31 ตัวอย่าง NB2 (ส่วนที่ 2)

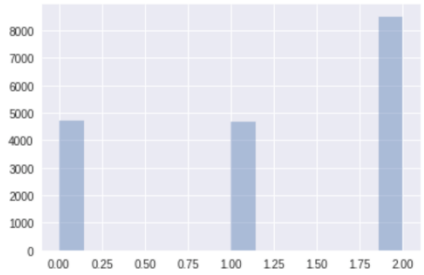
In [23]:
```
gt2 = tt1.time()

sns.distplot(y_crossval[['BUILDINGID']],kde=False)

n = n + 1
r0 = n, (tt1.time() - gt0)
r1 = n, (tt1.time() - gt1)
r2 = n, (tt1.time() - gt2)
r4 = tt1.time() - gt2
r5 = r5 + r4

%store r0 >> logs-total-time.csv
%store r1 >> logs-cell-delay-time.csv
%store r2 >> logs-cell-no-delay-time.csv
```

```
Writing 'r0' (tuple) to file 'logs-total-time.csv'.
Writing 'r1' (tuple) to file 'logs-cell-delay-time.csv'.
Writing 'r2' (tuple) to file 'logs-cell-no-delay-time.csv'.
```



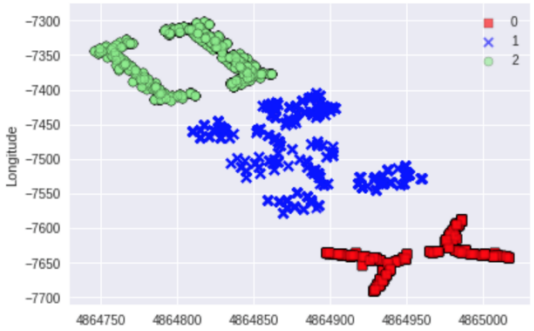
ภาพประกอบ 32 ตัวอย่าง NB3 (ส่วนที่ 1)

```
plt.xlabel('Latitude')
plt.ylabel('Longitude')
plt.legend(loc='upper right')
plt.tight_layout()

n = n + 1
r0 = n, (tt1.time() - gt0)
r1 = n, (tt1.time() - gt1)
r2 = n, (tt1.time() - gt2)
r4 = tt1.time() - gt2
r5 = r5 + r4

%store r0 >> logs-total-time.csv
%store r1 >> logs-cell-delay-time.csv
%store r2 >> logs-cell-no-delay-time.csv
```

```
Writing 'r0' (tuple) to file 'logs-total-time.csv'.
Writing 'r1' (tuple) to file 'logs-cell-delay-time.csv'.
Writing 'r2' (tuple) to file 'logs-cell-no-delay-time.csv'.
```



ภาพประกอบ 33 ตัวอย่าง NB3 (ส่วนที่ 2)

In [32]:
```
gt2 = tt1.time()

def nested_crossval(reg_list,reg_labels, model_param_grid=model_param_grid, model_scores = model_scores,
                    X = X_train, y= y_train, label_extension = None):
    '''
    Inputs:
    reg_model        : List of Regression model instances
    reg_label        : List of Regression model labels
    model_param_grid : List of parameter grids
    X                : explanatory variables
    y                : response variable array
    model_scores     : Dictionary to store nested cross-validation scores
    label_extension  : Extension to regression label in model_scores key

    Outputs:
    model_scores     : Updated dictionary of nested cross-validation scores
    '''


    for reg_model, reg_label in zip(reg_list, reg_labels):

        #print(param_grid)

        gs = (GridSearchCV(estimator=reg_model,
                           param_grid=model_param_grid[reg_label],
                           cv=2,
                           scoring = 'neg_mean_squared_error',
                           n_jobs = 1))

        scores = cross_val_score(estimator=gs,
                                 X=X,
                                 y=y,
                                 cv=5,
                                 scoring='neg_mean_squared_error')
        scores = np.sqrt(np.abs(scores))
```

ภาพประกอบ 34 ตัวอย่าง NBTest (ส่วนที่ 1)

In [14]:
```
gt2 = tt1.time()

X_pca_crossval = pd.read_csv("data/X_pca_crossval.csv",index_col=0)
y_crossval = pd.read_csv("data/y_crossval.csv",index_col=0)

X_pca_holdout = pd.read_csv("data/X_pca_holdout.csv",index_col=0)
y_holdout = pd.read_csv("data/y_holdout.csv",index_col=0)

X_raw_crossval = pd.read_csv("data/X_raw_crossval.csv",index_col=0)
X_raw_holdout = pd.read_csv("data/X_raw_holdout.csv",index_col=0)

X_pca_crossval.shape,y_crossval.shape,X_pca_holdout.shape,y_holdout.shape

n = n + 1
r0 = n, (tt1.time() - gt0)
r1 = n, (tt1.time() - gt1)
r2 = n, (tt1.time() - gt2)
r4 = tt1.time() - gt2
r5 = r5 + r4

%store r0 >> logs-total-time.csv
%store r1 >> logs-cell-delay-time.csv
%store r2 >> logs-cell-no-delay-time.csv
```

```
Writing 'r0' (tuple) to file 'logs-total-time.csv'.
Writing 'r1' (tuple) to file 'logs-cell-delay-time.csv'.
Writing 'r2' (tuple) to file 'logs-cell-no-delay-time.csv'.
```

ภาพประกอบ 35 ตัวอย่าง NBTest (ส่วนที่ 2)

ประวัติผู้เขียน

ชื่อ-สกุล	ปริวรรต ปธานราษฎร์
วัน เดือน ปี เกิด	14 Nov 1990
สถานที่เกิด	จันทบุรี
วุฒิการศึกษา	มหาวิทยาลัยศรีนครินทรวิโรฒ
ที่อยู่ปัจจุบัน	104 ถนนศาลาตาผ่อง ตำบลทางเกวียน อำเภอแกลง จังหวัดระยอง