



โครงการงาน

Final Assignment

จัดทำโดย

นายณัฐ	ชูกำแพง	รหัสนักศึกษา 60070134
นางสาวณัฐนิชา	ชัยศิริพานิช	รหัสนักศึกษา 60070135
นายพนทกร	มาศิรี	รหัสนักศึกษา 60070144

เสนอ

รศ.ดร.กิตติ์สุชาติ พสุภา

โครงการนี้เป็นส่วนหนึ่งของวิชา MACHINE LEARNING (06026125)

ภาคเรียนที่ 2 ปีการศึกษา 2562

สาขาวิทยาการข้อมูลและการวิเคราะห์ทางธุรกิจ คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

คำนำ

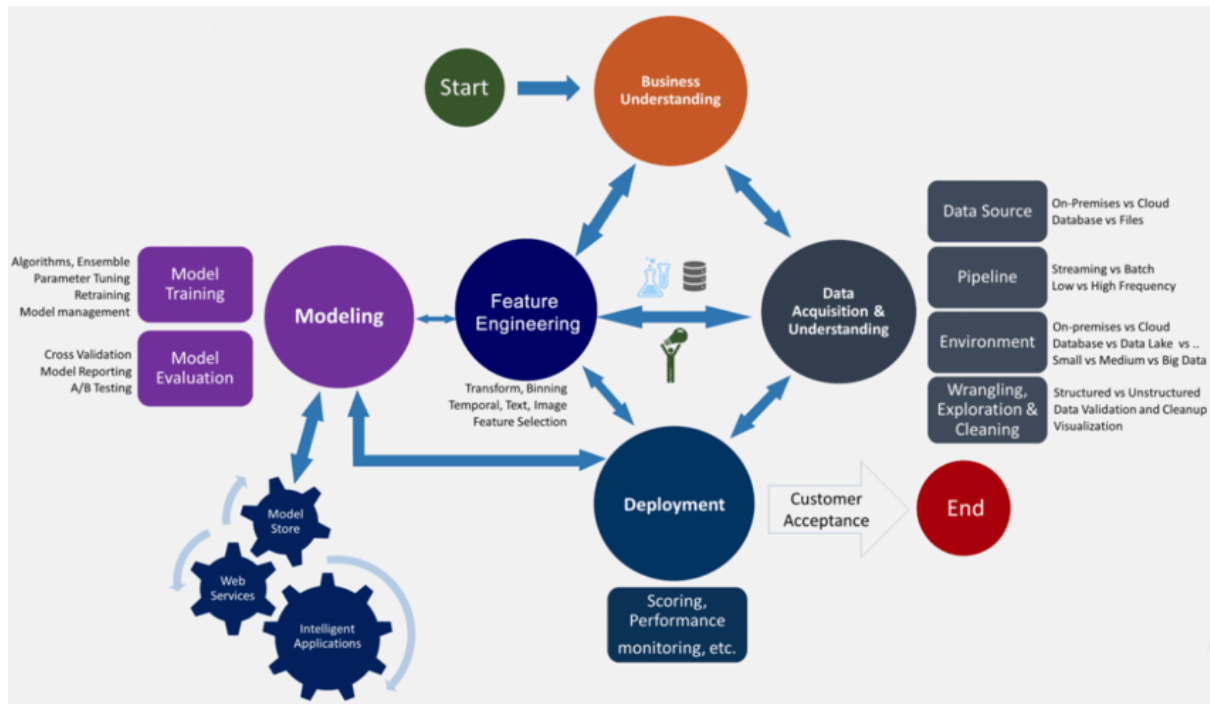
โครงงานนี้เป็นส่วนหนึ่งของวิชา MACHINE LEARNING (06026125) โดยมีจุดประสงค์เพื่อการพัฒนาผลทางการ ศึกษาตลอดระยะเวลาหนึ่งเทอมที่ผ่านมาในวิชานี้ แทนการสอบวัดผลทางการศึกษา (Final Exam) ภายใน โครงงานนี้ใช้ภาษา Python ในการวิเคราะห์และแสดงผลในรูปแบบต่าง ๆ ตามโจทย์ที่ได้ รับมอบหมาย รายงานเล่มนี้จัดทำเพื่อเป็นการอธิบายแต่ละส่วน วิธีการดำเนินงาน กระบวนการในการแก้ปัญหา ผลลัพธ์ที่ได้ในแต่ละกระบวนการ รวมถึงหลักการต่าง ๆ ที่ได้นำมาประยุกต์ใช้ในการทำโครงงาน

คณะผู้จัดทำ

สารบัญ

เรื่อง	หน้า
วิธีการดำเนินงาน	1
1. Classification	2-3
1.1 Exploration	3-8
1.2 Data preparation	9
1.3 Modeling	10-15
1.4 สรุปผลการทดลอง	15
2. Regression	16
2.1 Exploration	16-17
2.2 Feature Engineering	17-28
2.3 สรุปผลการทดลอง	28
3. Clustering	29-30
3.1 Exploration	30-33
3.2 Data preparation	34
3.3 Modeling	35-36
3.4 สรุปผลการทดลอง	37
References	38

วิธีการดำเนินการ



(1)

ทางคณะผู้จัดทำได้ดำเนินการแก้ปัญหาโดยทำตาม process การดำเนินการของ data science

ดังภาพด้านบน ซึ่งประกอบไปด้วย Process หลักๆอยู่ 5 ส่วน ดังนี้

- **Business Understanding** คือ การเข้าใจในปัญหาที่เรา กำลังจะทำ
- **Data Acquisition & Understanding** คือการได้มาซึ่งข้อมูลและการเข้าใจในข้อมูลที่ใช้ในการวิเคราะห์
- **Feature Engineering** คือ การนำข้อมูลมาจัดการให้มีประสิทธิภาพเพื่อนำไปทำแบบจำลอง
- **Modeling** คือ การสร้างแบบจำลองเพื่อตอบโจทย์ปัญหาของเรา
- **Deployment** คือ การนำเอาแบบจำลองที่สำเร็จนำไปใช้จริง

การแก้ปัญหา

1. Classification

ใช้ข้อมูล Heart Disease Classification เป็นข้อมูลของคนไข้ จำนวน 303 คน แต่ละคนมี

คุณลักษณะอธิบายจำนวน 13 Factor ซึ่งแบ่งเป็น 2 ประเภท คือ เป็นโรค และ ไม่เป็นโรค

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

โดยเป้าหมายคือ การทำ Classification ว่าเป็นโรคหัวใจหรือไม่ โดยใช้ข้อมูลที่มีมาให้ นำมาวิเคราะห์

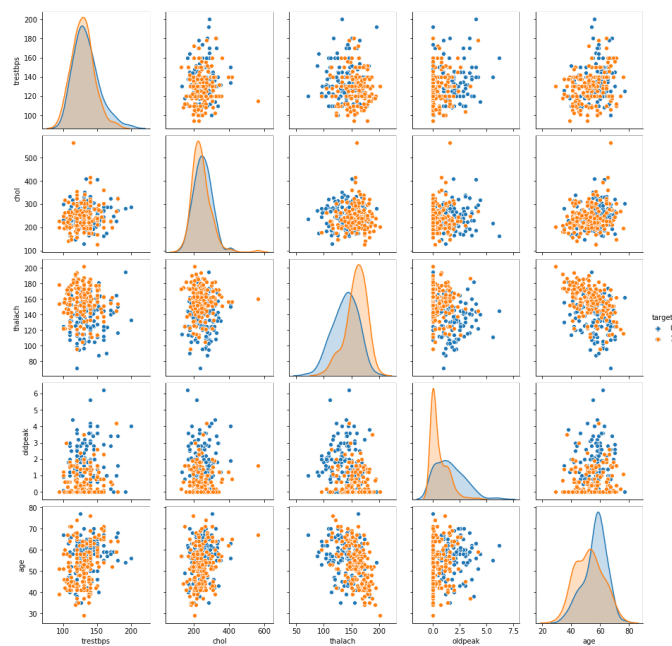
โดยแต่ละ Factor มีคำอธิบาย ดังนี้

- age คือ อายุของคนไข้
- sex คือ เพศของคนไข้ โดยที่ 0 คือ เพศหญิง และ 1 คือ เพศชาย
- cp (Chest pain type) คือ ประเภทของการเจ็บหน้าอก มี 4 ค่า
- trestbps (Resting blood pressure) คือ ความดันโลหิตขณะที่คนไข้กำลังพัก มีหน่วยเป็น mm Hg
- chol (Serum cholesterol in mg/dl) คือ ระดับของเซรัมคอเลสเตอรอลของคนไข้ มีหน่วยเป็น (mg / dL มิลลิกรัมต่อเดซิลิตร)
- fbs คือ ระดับค่าน้ำตาลในเลือดมากกว่า 120 mg/dl หรือไม่ โดยที่ 1 คือ True (มากกว่า 120 mg/dl) และ 0 คือ False (น้อยกว่า 120 mg/dl)
- restecg คือ ผลลัพธ์ของคลื่นไฟฟ้าของหัวใจขณะที่คนไข้กำลังพัก มี 3 ค่า คือ 0,1,2
- thalach คือ อัตราการเต้นของหัวใจสูงสุดของคนไข้
- exang คือ โรคหลอดเลือดหัวใจตีบที่เกิดจากการออกกำลังกาย โดยที่ 0 คือ ไม่ใช่ และ 1 คือ ใช่
- oldpeak คือ ST depression ที่เกิดจากการออกกำลังกายและความสัมพันธ์กับการพักผ่อน

- slope คือ ความชันสูงสุดของส่วนการออกกำลังกาย ST segment โดยมี 3 ค่า ค่าที่ 1 คือ upsloping ค่าที่ 2 คือ flat และค่าที่ 3 downsloping
- ca คือ จำนวนเส้นเลือดที่เกิดจากการทำ flu
- thal คือ ความผิดปกติของเลือดที่เรียกว่าธาตุซีเมีย
- Target คือ คนไข้เป็นโรคหัวใจหรือไม่ โดยที่ 0 คือ ไม่เป็น และ 1 คือ เป็น

1.1. Exploration

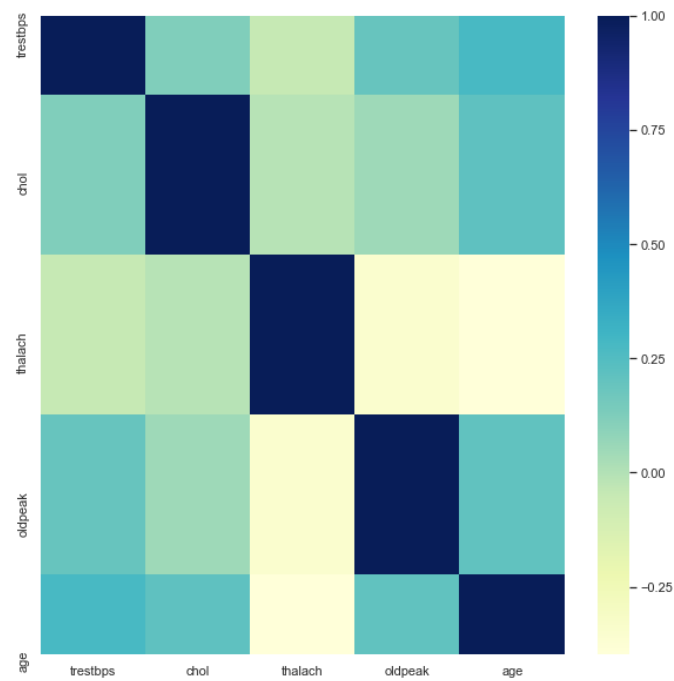
- ดูว่าข้อมูลมีลักษณะและมีความสัมพันธ์เป็นอย่างไร ด้วยการใช้เทคนิคการนำเสนอข้อมูลด้วยรูปแบบต่างๆ



จากภาพข้างต้นจะแสดงให้เห็นถึงความสัมพันธ์ระหว่างคนที่ เป็นโรคกับไม่เป็นโรค

ในตัวแปรประเภท Numerical

- Heat - map

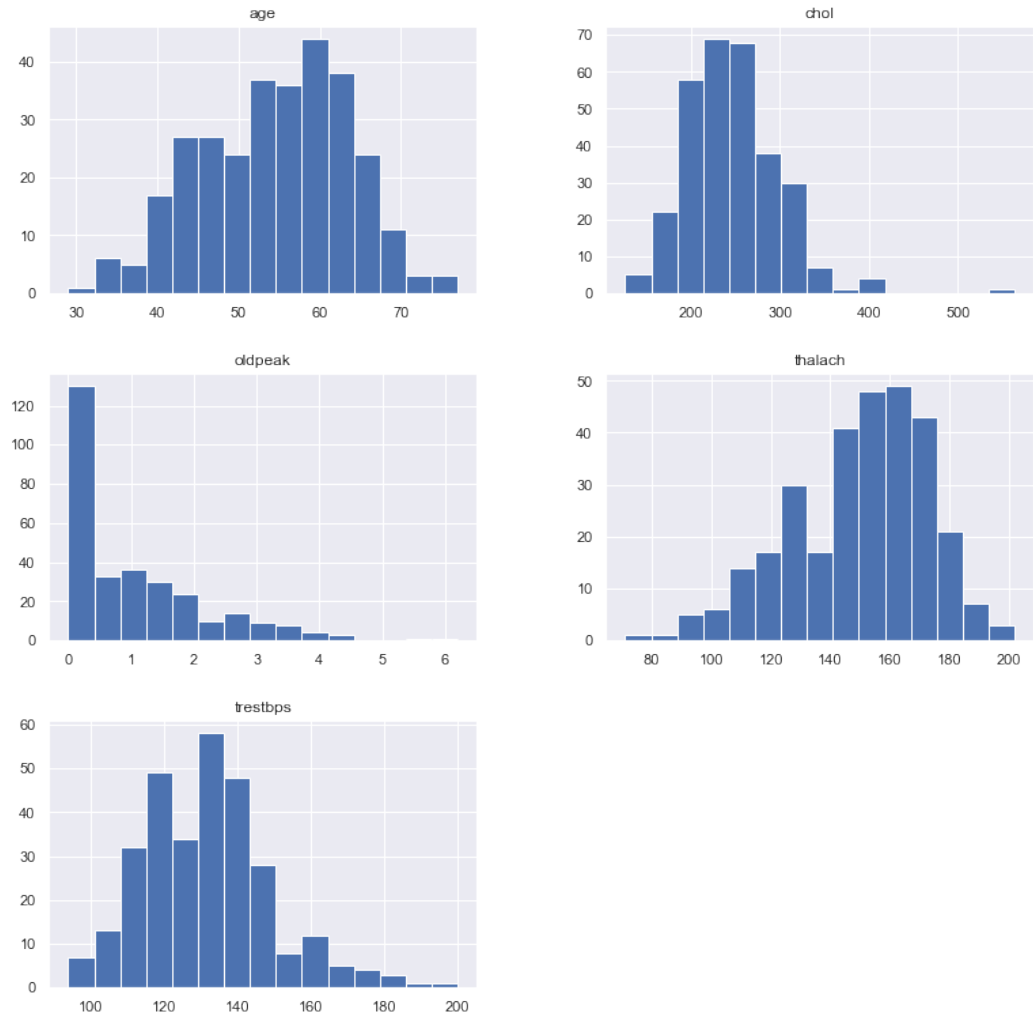


จากภาพข้างต้นจะแสดงความสัมพันธ์ระหว่างตัวแปรประเภท Numerical

ว่ามีความสัมพันธ์กันมากน้อยเท่าใด

- Histogram

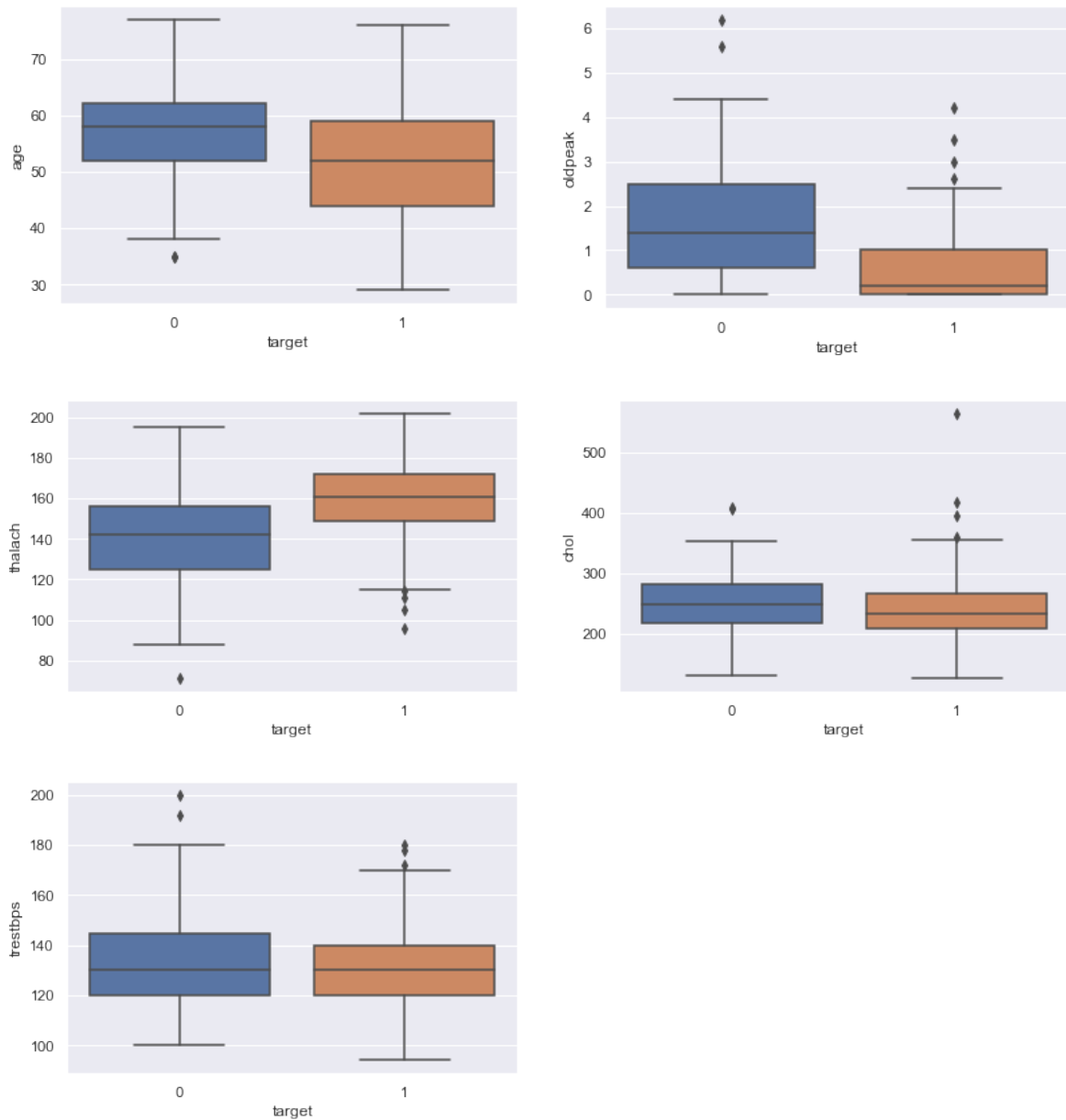
Histogram for each numeric input variable



จากภาพข้างต้นจะแสดงการกระจายของข้อมูลในแต่ละตัวแปรประเภท

Numerical

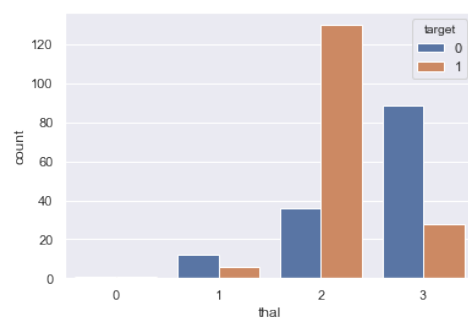
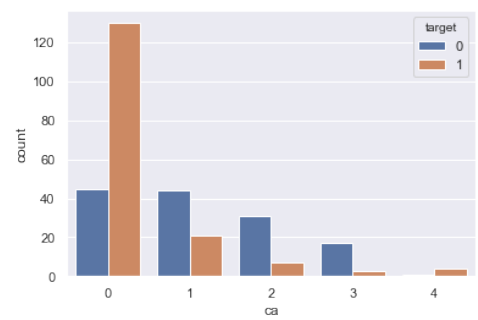
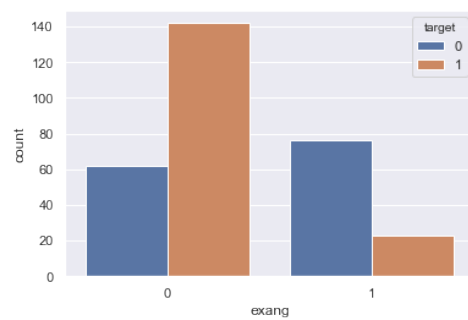
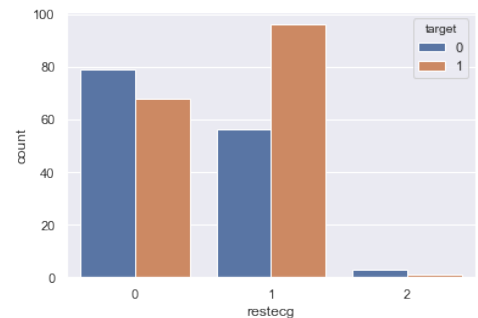
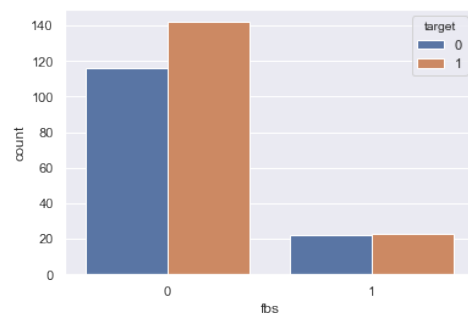
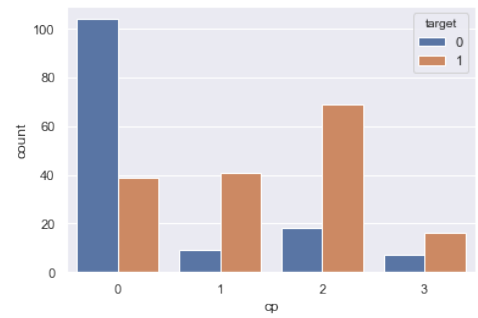
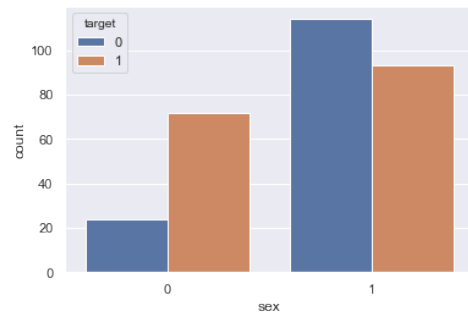
● Box-Plot for each target



จากภาพข้างต้นจะแสดงการกระจายตัวของข้อมูลที่เป็นประเภท Numerical

เปรียบเทียบกับ Target ว่าเป็นโรคหรือไม่เป็นโรค

- Categorical Features



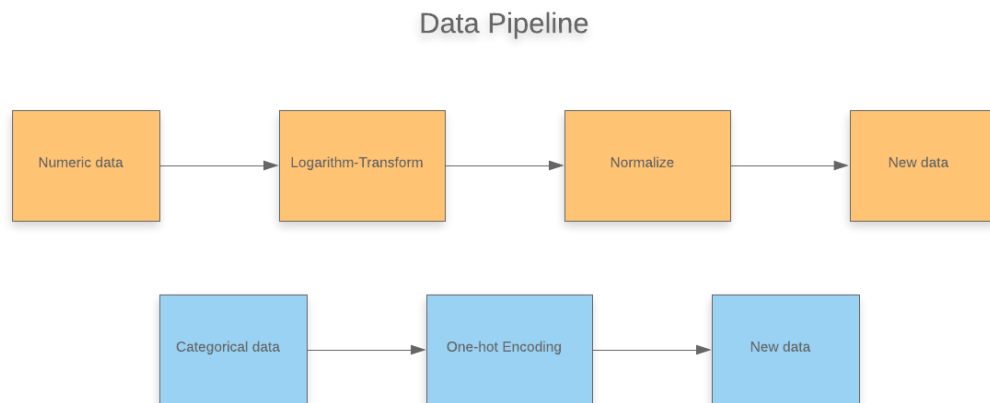
- Feature Importances

	score
0	
cp	0.229090
thal	0.180093
ca	0.138294
exang	0.088115
oldpeak	0.065899
slope	0.059119
sex	0.051821
age	0.038006
fbs	0.034018
restecg	0.029869
thalach	0.029622
trestbps	0.029416
chol	0.026637

จากภาพข้างต้นเป็นการแสดงความสำคัญในแต่ละตัวแปรที่มีผลต่อการทำ

Classification

1.2. Data preparation



ทางคณะผู้จัดทำได้ออกแบบวิธีการ Preprocess และ Feature Engineering กับ ชุดข้อมูล โดยแบ่งออกเป็น 2 กลุ่ม ได้กระบวนการดังภาพข้างต้น

สำหรับ Numerical data เนื่องจากข้อมูลมีการกระจายที่มากรวมถึงมี outlier ปะปน มาด้วยทำให้การใช้ Logarithm-Transform จะช่วยทำให้ข้อมูลกระจายน้อยลง เพียงแค่นั้น ยังไม่พอ เนื่องจากข้อมูลมีช่วงที่ไม่เท่ากันเราเลยใช้ Normalization ในการปรับให้ชุดข้อมูลอยู่ในช่วงเดียวกัน

สำหรับ Categorical data เหตุผลในการเลือกใช้ One-hot Encoding เนื่องจากถ้าเราใช้ Label Encoding จะทำให้มีข้อเสียสำหรับ model ที่เป็น linear เหตุเพราะ เช่นเราแปลงเพศชาย ให้เป็น 1 และหญิงให้เป็น 0 มันแสดงให้เห็นว่า 1 นั้นมากกว่า 0

1.3. Modeling

หลังจากเตรียมข้อมูลเรียบร้อยแล้วเราได้ทำการทดสอบประสิทธิภาพของ model เพื่อใช้เป็น Baseline โดยเราได้ทำการทดลองด้วย K-Fold Cross-Validation แยกเป็น 3 แบบที่ใช้ข้อมูล Numeric, Categorical, Numeric + Categorical ได้ผลดังนี้

	model	acc (cv=10)	f1 (cv=10)	precision (cv=10)	recall (cv=10)
0	SVC	0.672473	0.507300	0.563158	0.479140
1	XGBClassifier	0.616882	0.474233	0.573333	0.415215
2	LogisticRegression	0.643333	0.485049	0.570588	0.440000
3	KNeighborsClassifier	0.606559	0.469896	0.562500	0.416559
4	DecisionTreeClassifier	0.593441	0.456584	0.564706	0.401774
5	RandomForestClassifier	0.626989	0.467590	0.566667	0.416989
6	GradientBoostingClassifier	0.607097	0.464587	0.570588	0.410430

(Numeric data)

	model	acc (cv=10)	f1 (cv=10)	precision (cv=10)	recall (cv=10)
0	SVC	0.788387	0.529896	0.568750	0.503387
1	XGBClassifier	0.761828	0.512293	0.571429	0.471828
2	LogisticRegression	0.821290	0.538077	0.573333	0.512957
3	KNeighborsClassifier	0.771828	0.525706	0.568750	0.496828
4	DecisionTreeClassifier	0.696237	0.476899	0.564706	0.427903
5	RandomForestClassifier	0.765269	0.519482	0.568750	0.486935
6	GradientBoostingClassifier	0.761398	0.522102	0.573333	0.489731

(Categorical data)

	model	acc (cv=10)	f1 (cv=10)	precision (cv=10)	recall (cv=10)
0	SVC	0.788172	0.529274	0.568750	0.503172
1	XGBClassifier	0.771720	0.519006	0.571429	0.481720
2	LogisticRegression	0.811290	0.533461	0.573333	0.506290
3	KNeighborsClassifier	0.764731	0.544697	0.566667	0.534731
4	DecisionTreeClassifier	0.718817	0.500624	0.564706	0.463817
5	RandomForestClassifier	0.775054	0.517203	0.564706	0.490054
6	GradientBoostingClassifier	0.761720	0.522534	0.568750	0.493387

(Numeric + Categorical)

จากผลที่ได้นั้นจะทำให้เราเห็นว่า การใช้ข้อมูลแค่ Numeric ทำให้ประสิทธิภาพของ model แย่กว่าการที่ใช้ร่วมกับ Categorical และยังทำให้เห็นอีกว่าเพียงแค่ Categorical data ก็ทำให้ model มีประสิทธิภาพที่ดีได้

ผลลัพธ์ที่ได้นั้นทำให้เราเลือก model มา 3 ตัวเพื่อใช้ในการทำ Hyperparameter Tuning โดย model ที่เราเลือกนั้นได้แก่ Support vector machine, K-Nearest Neighbor, Logistic Regression

โดยเทคนิคที่เราใช้ในการทำ Hyperparameter Tuning ก็คือ Bayesian Optimization ซึ่งเป็นวิธีการที่นำเอา Probabilistic Model ในการ หาค่าแทนที่เราจะสุ่มมั่วๆ หรือใช้ Grid ในการค้นหา เราก็ random จาก Prior ปัจจุบัน แล้วทำการ update Posterior แล้ววัดผลแบบวิธีก่อนหน้านี้ (2)

SVC	<p>Accuracy on testset: 0.9016393442622951 F1 on testset: 0.8999999999999999 Precision on testset: 0.84375 Recall on testset: 0.9642857142857143</p> <table><thead><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr></thead><tbody><tr><td>0</td><td>0.97</td><td>0.85</td><td>0.90</td><td>33</td></tr><tr><td>1</td><td>0.84</td><td>0.96</td><td>0.90</td><td>28</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.90</td><td>61</td></tr><tr><td>macro avg</td><td>0.90</td><td>0.91</td><td>0.90</td><td>61</td></tr><tr><td>weighted avg</td><td>0.91</td><td>0.90</td><td>0.90</td><td>61</td></tr></tbody></table> <p>[[28 5] [1 27]]</p>		precision	recall	f1-score	support	0	0.97	0.85	0.90	33	1	0.84	0.96	0.90	28	accuracy			0.90	61	macro avg	0.90	0.91	0.90	61	weighted avg	0.91	0.90	0.90	61	<table><thead><tr><th></th><th>model</th><th>acc (cv=10)</th><th>f1 (cv=10)</th><th>precision (cv=10)</th><th>recall (cv=10)</th></tr></thead><tbody><tr><td>0</td><td>SVC</td><td>0.817667</td><td>0.833096</td><td>0.807701</td><td>0.867596</td></tr></tbody></table>		model	acc (cv=10)	f1 (cv=10)	precision (cv=10)	recall (cv=10)	0	SVC	0.817667	0.833096	0.807701	0.867596
	precision	recall	f1-score	support																																								
0	0.97	0.85	0.90	33																																								
1	0.84	0.96	0.90	28																																								
accuracy			0.90	61																																								
macro avg	0.90	0.91	0.90	61																																								
weighted avg	0.91	0.90	0.90	61																																								
	model	acc (cv=10)	f1 (cv=10)	precision (cv=10)	recall (cv=10)																																							
0	SVC	0.817667	0.833096	0.807701	0.867596																																							
KNN	<p>Accuracy on testset: 0.8360655737704918 F1 on testset: 0.8437499999999999 Precision on testset: 0.75 Recall on testset: 0.9642857142857143</p> <table><thead><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr></thead><tbody><tr><td>0</td><td>0.96</td><td>0.73</td><td>0.83</td><td>33</td></tr><tr><td>1</td><td>0.75</td><td>0.96</td><td>0.84</td><td>28</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.84</td><td>61</td></tr><tr><td>macro avg</td><td>0.85</td><td>0.85</td><td>0.84</td><td>61</td></tr><tr><td>weighted avg</td><td>0.86</td><td>0.84</td><td>0.84</td><td>61</td></tr></tbody></table> <p>[[24 9] [1 27]]</p>		precision	recall	f1-score	support	0	0.96	0.73	0.83	33	1	0.75	0.96	0.84	28	accuracy			0.84	61	macro avg	0.85	0.85	0.84	61	weighted avg	0.86	0.84	0.84	61	<table><thead><tr><th></th><th>model</th><th>acc (cv=10)</th><th>f1 (cv=10)</th><th>precision (cv=10)</th><th>recall (cv=10)</th></tr></thead><tbody><tr><td>0</td><td>KNeighborsClassifier</td><td>0.817833</td><td>0.836632</td><td>0.830615</td><td>0.853248</td></tr></tbody></table>		model	acc (cv=10)	f1 (cv=10)	precision (cv=10)	recall (cv=10)	0	KNeighborsClassifier	0.817833	0.836632	0.830615	0.853248
	precision	recall	f1-score	support																																								
0	0.96	0.73	0.83	33																																								
1	0.75	0.96	0.84	28																																								
accuracy			0.84	61																																								
macro avg	0.85	0.85	0.84	61																																								
weighted avg	0.86	0.84	0.84	61																																								
	model	acc (cv=10)	f1 (cv=10)	precision (cv=10)	recall (cv=10)																																							
0	KNeighborsClassifier	0.817833	0.836632	0.830615	0.853248																																							
Logistic Reg.	<p>Accuracy on testset: 0.9016393442622951 F1 on testset: 0.8999999999999999 Precision on testset: 0.84375 Recall on testset: 0.9642857142857143</p> <table><thead><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr></thead><tbody><tr><td>0</td><td>0.97</td><td>0.85</td><td>0.90</td><td>33</td></tr><tr><td>1</td><td>0.84</td><td>0.96</td><td>0.90</td><td>28</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.90</td><td>61</td></tr><tr><td>macro avg</td><td>0.90</td><td>0.91</td><td>0.90</td><td>61</td></tr><tr><td>weighted avg</td><td>0.91</td><td>0.90</td><td>0.90</td><td>61</td></tr></tbody></table> <p>[[28 5] [1 27]]</p>		precision	recall	f1-score	support	0	0.97	0.85	0.90	33	1	0.84	0.96	0.90	28	accuracy			0.90	61	macro avg	0.90	0.91	0.90	61	weighted avg	0.91	0.90	0.90	61	<table><thead><tr><th></th><th>model</th><th>acc (cv=10)</th><th>f1 (cv=10)</th><th>precision (cv=10)</th><th>recall (cv=10)</th></tr></thead><tbody><tr><td>0</td><td>LogisticRegression</td><td>0.830167</td><td>0.841212</td><td>0.839939</td><td>0.851363</td></tr></tbody></table>		model	acc (cv=10)	f1 (cv=10)	precision (cv=10)	recall (cv=10)	0	LogisticRegression	0.830167	0.841212	0.839939	0.851363
	precision	recall	f1-score	support																																								
0	0.97	0.85	0.90	33																																								
1	0.84	0.96	0.90	28																																								
accuracy			0.90	61																																								
macro avg	0.90	0.91	0.90	61																																								
weighted avg	0.91	0.90	0.90	61																																								
	model	acc (cv=10)	f1 (cv=10)	precision (cv=10)	recall (cv=10)																																							
0	LogisticRegression	0.830167	0.841212	0.839939	0.851363																																							

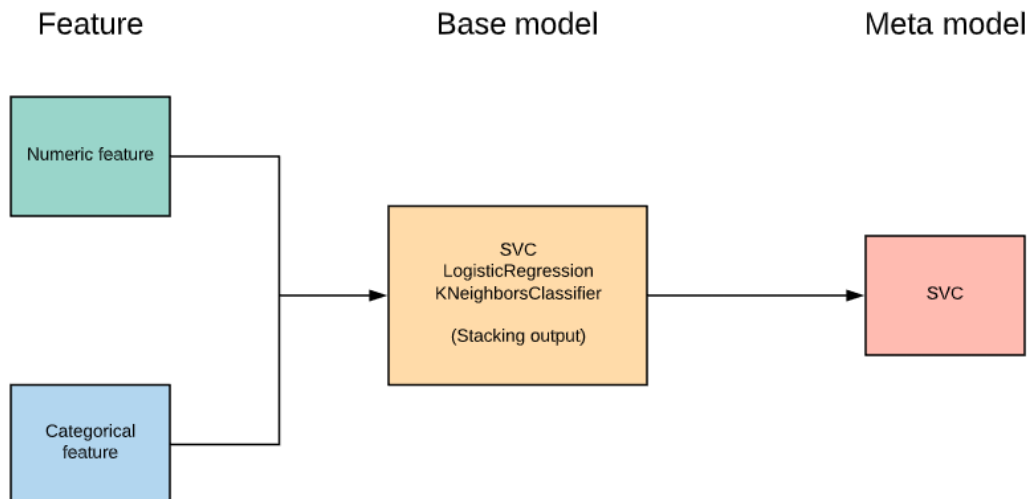
จะเห็นได้ว่าหลังจากที่ทำการ Tuning ประสิทธิภาพของ model เพิ่มขึ้นอย่างเห็นได้ชัดเจนรวม

ถึงเราได้เอาไปทดสอบในชุด test set ด้วยผลลัพธ์ที่ได้อยู่ข้างต้นด้วยแล้ว โดย model ที่ให้ดีที่สุดก็คือ

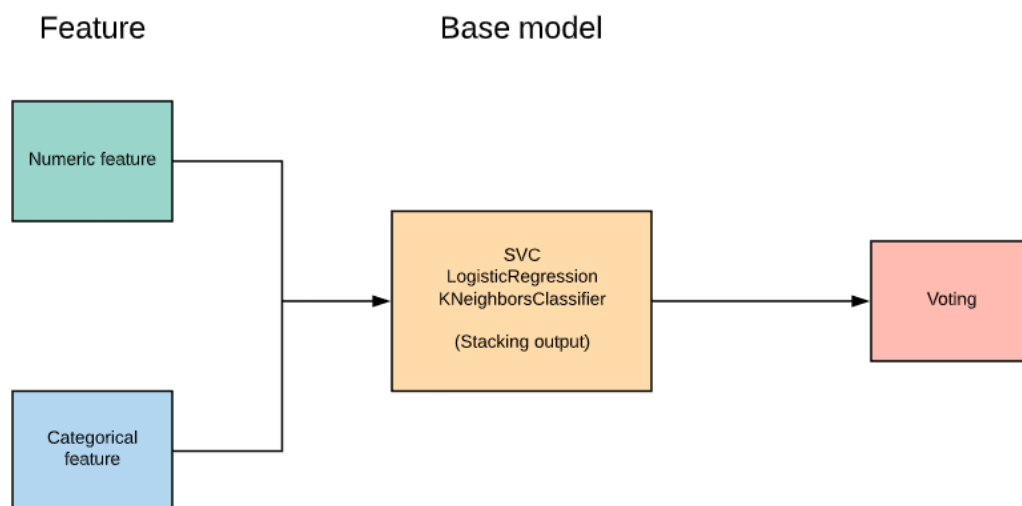
Logistic Regression

เพื่อเพิ่มประสิทธิภาพจากเดิม เราได้ทำการออกแบบ stacking model โดยเราได้ออกแบบทั้งหมด 3 แบบดังนี้

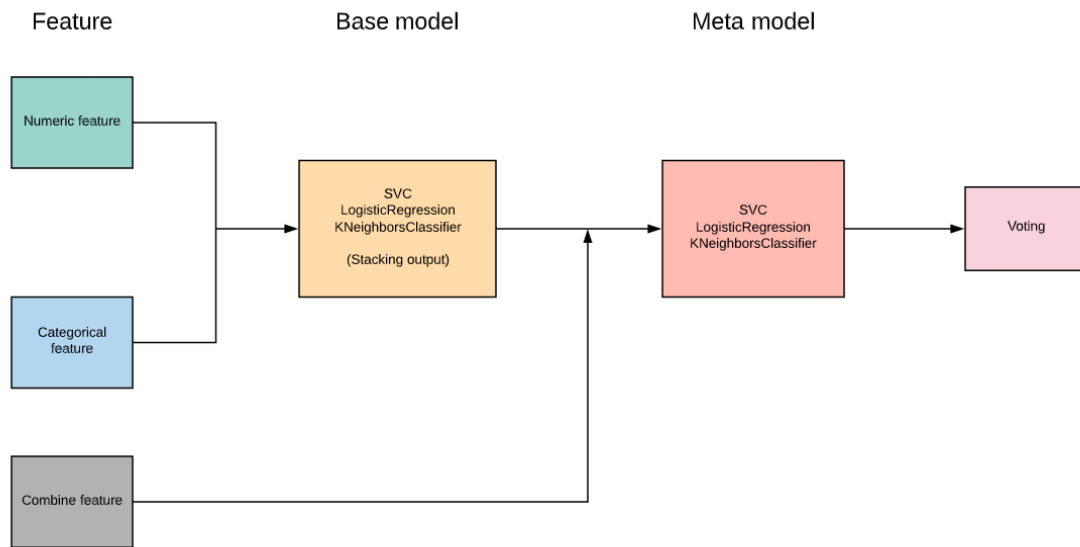
- Simple Stacking



- Majority Vote



- Stacking + Majority vote with multi-meta-learner



ซึ่งได้ผลลัพธ์ดังต่อไปนี้

model	acc (cv=10)	f1 (cv=10)	precision (cv=10)	recall (cv=10)
0 Stacking 3 meta-learner and Majority vote	0.838667	0.85937	0.847083	0.883516

Accuracy on testset: 0.9016393442622951
 F1 on testset: 0.8999999999999999
 Precision on testset: 0.84375
 Recall on testset: 0.9642857142857143

	precision	recall	f1-score	support
0	0.97	0.85	0.90	33
1	0.84	0.96	0.90	28
accuracy			0.90	61
macro avg	0.90	0.91	0.90	61
weighted avg	0.91	0.90	0.90	61

[[28 5]
[1 27]]

model	acc (cv=10)	f1 (cv=10)	precision (cv=10)	recall (cv=10)
0 Ensemble majority vote	0.830333	0.851836	0.835984	0.875824

Accuracy on testset: 0.9016393442622951
 F1 on testset: 0.8999999999999999
 Precision on testset: 0.84375
 Recall on testset: 0.9642857142857143

	precision	recall	f1-score	support
0	0.97	0.85	0.90	33
1	0.84	0.96	0.90	28
accuracy			0.90	61
macro avg	0.90	0.91	0.90	61
weighted avg	0.91	0.90	0.90	61

[[28 5]
[1 27]]

model	acc (cv=10)	f1 (cv=10)	precision (cv=10)	recall (cv=10)
0 Stacking 1 meta-learner	0.834667	0.855429	0.846131	0.876374

Accuracy on testset: 0.9016393442622951
 F1 on testset: 0.8999999999999999
 Precision on testset: 0.84375
 Recall on testset: 0.9642857142857143

	precision	recall	f1-score	support
0	0.97	0.85	0.90	33
1	0.84	0.96	0.90	28
accuracy			0.90	61
macro avg	0.90	0.91	0.90	61
weighted avg	0.91	0.90	0.90	61

[[28 5]
[1 27]]

ทำให้เห็นว่าผลลัพธ์จากการทำ Stacking model ให้ผลลัพธ์ในเชิง Generalize ที่ดีกว่าไม่ว่าจะค่า score ใดๆในตอนทำ validation โดยรูปแบบที่ได้ผลลัพธ์ที่ดีที่สุดคือ Stacking + Majority Vote แต่จากผลลัพธ์ใน testset สำหรับค่า score ต่างๆจะได้เท่ากันหมดเลย

1.4. สรุปผลการทดลอง

ผลลัพธ์หลังจากทำการทดลองทำให้เราสร้าง model ที่มีประสิทธิภาพสูงสุดก็คือการทำ Stacking + Majority Vote เป็นการใช้ความหลากหลายในการทำนายข้อมูลเพียงแค่ 303 ข้อมูล โดยค่าที่เราจะให้ความสนใจที่สุดคือ Recall เนื่องจากปัญหาของเรามี Sensitive case เราต้องการทำนายคนที่เป็นโรคแล้วเราบอกว่าเป็นโรคให้ดีที่สุด ซึ่งใน validation เราได้อยู่ที่ 88% และใน testset อยู่ที่ 96%

2. Regression

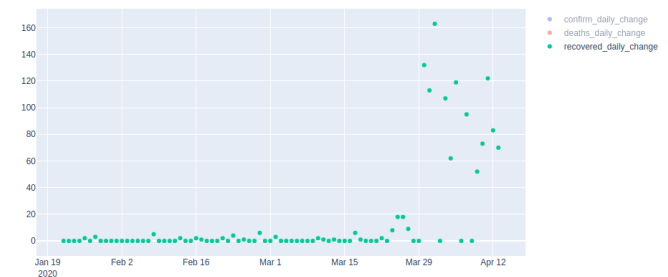
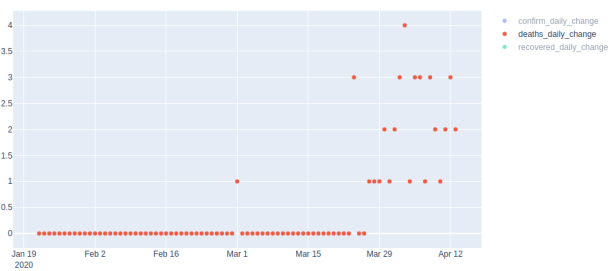
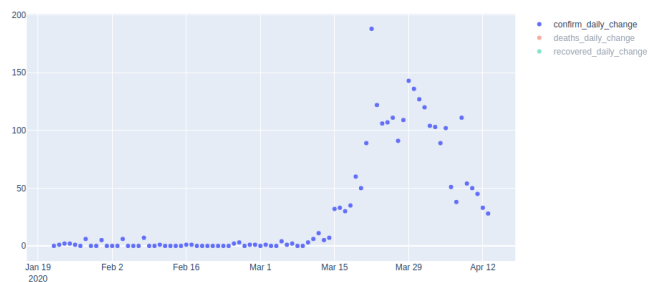
ข้อมูลของผู้ป่วยโรค COVID-19 ในประเทศไทย ตั้งแต่วันที่มีการ Outbreak ของเชื้อโรค

22/01/2020 จนถึง 13/04/2020

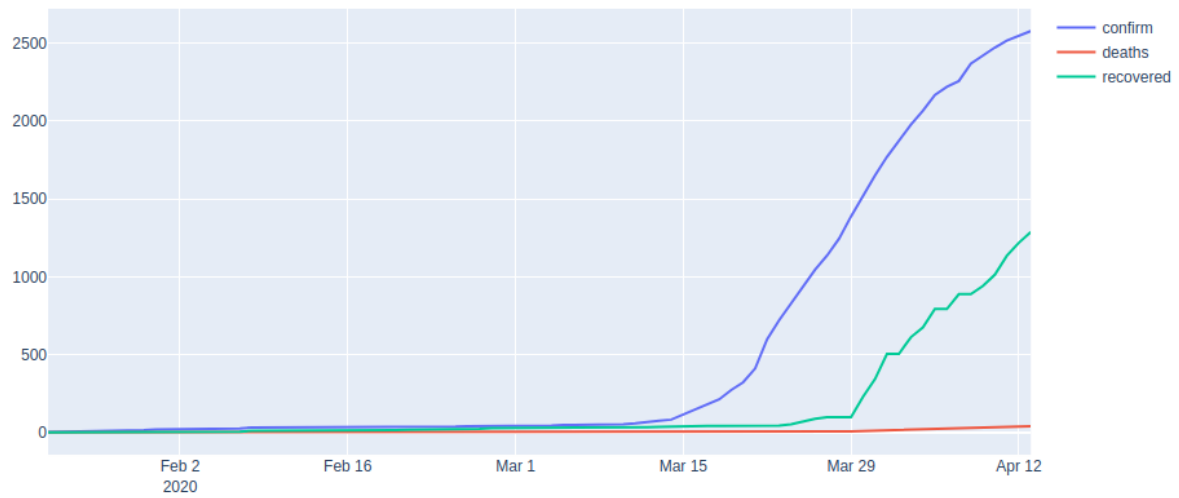
	confirm	deaths	recovered
date			
2020-01-22	2	0	0
2020-01-23	3	0	0
2020-01-24	5	0	0
2020-01-25	7	0	0
2020-01-26	8	0	2

2.1. Exploration

- จำนวนการเกิดเพิ่มขึ้น, การตาย, รักษา ได้ในแต่ละวัน



- จำนวนเคสสะสม



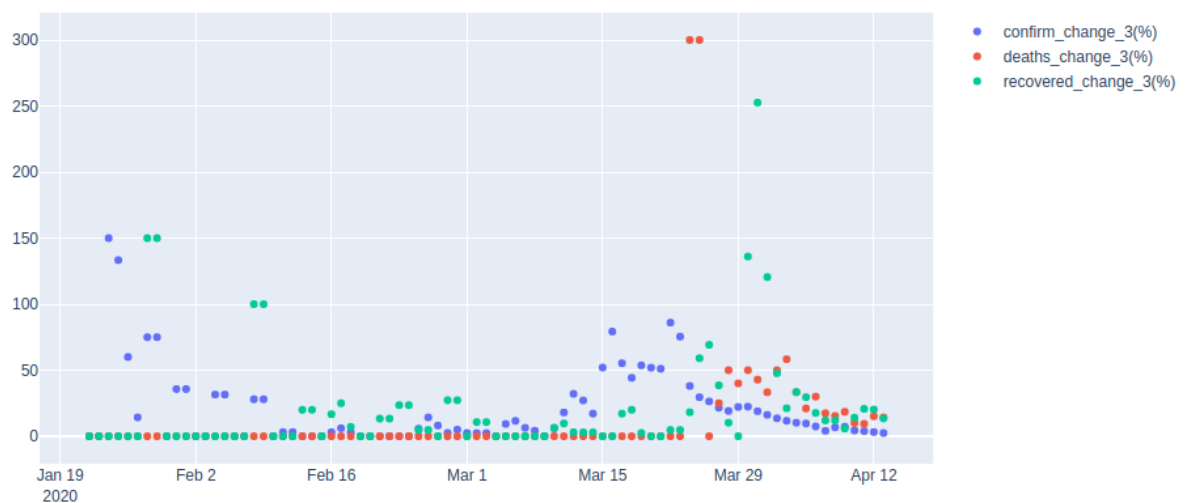
2.2. Feature Engineering

- Percentage of Change every N-days

เป็นการดูว่า จากวันปัจจุบันเทียบกับวันก่อนหน้า N วัน มีค่าเพิ่มเป็นกี่เปอร์เซ็นต์ จากภาพด้านล่าง

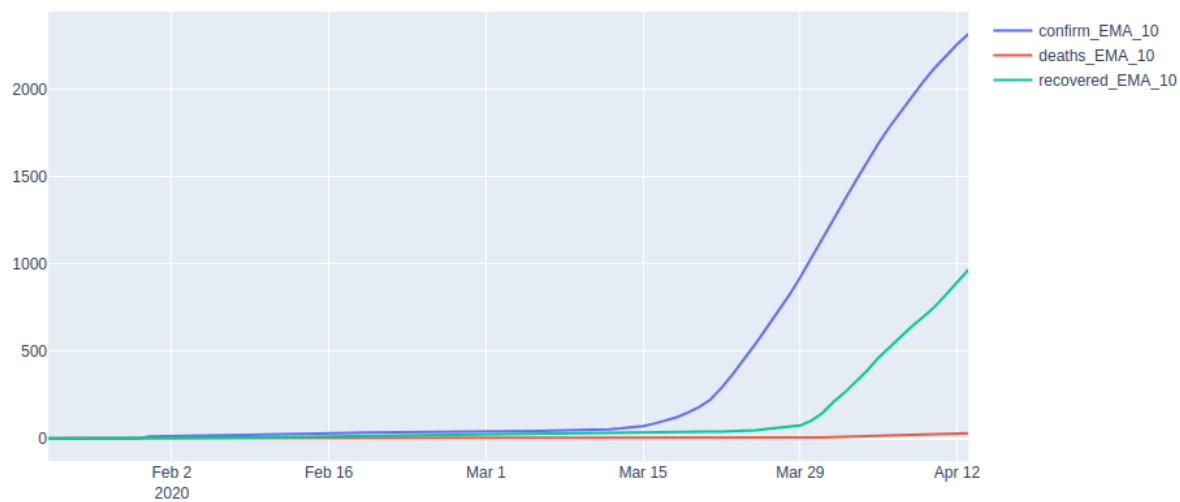
แสดงตัวอย่าง N=3

- Moving Average size=N



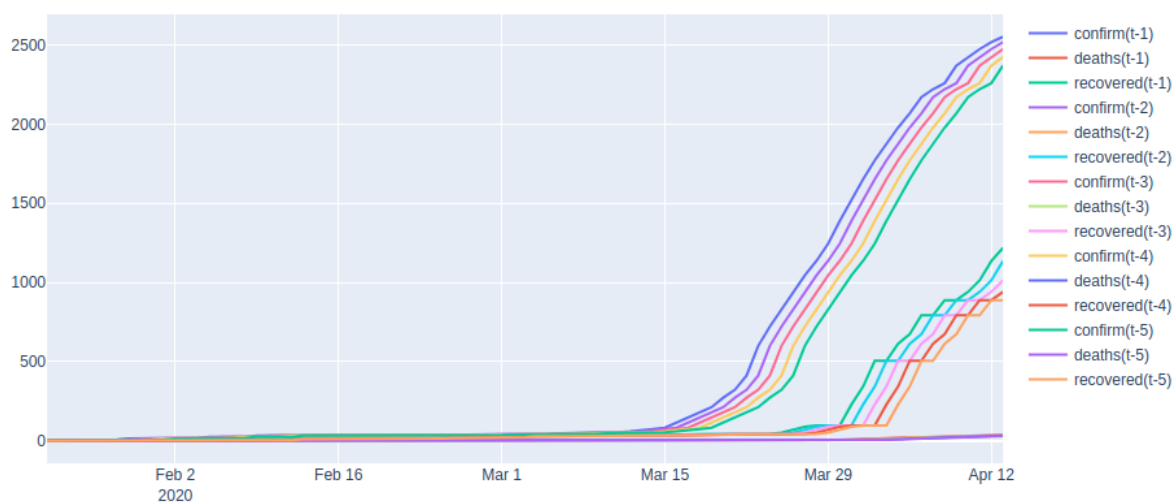
เราได้ใช้ Exponential moving average ในการ smooth ค่าความถี่สะสม โดยในตัวอย่างรูปเราใช้

$N=10$



● Lags N-days

เป็น feature ที่เอาวันก่อนหน้าวันปัจจุบัน N วันมาเป็น feature เพิ่ม ในตัวอย่างเราให้ $N=5$



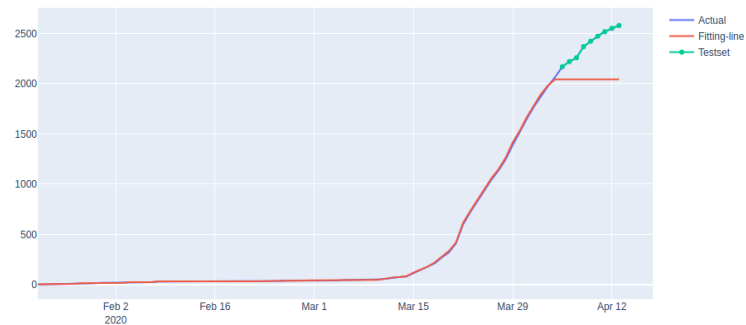
Modeling

เราได้ทำการแบ่งการทดลองออกเป็น 6 แบบ โดยเริ่มจาก

- Timestep only

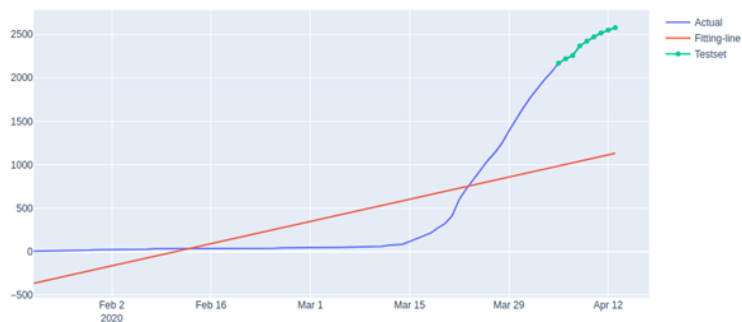
เป็นการนำเอาเฉพาะ Timestep มาเป็น feature เท่านั้น

Tree-based model



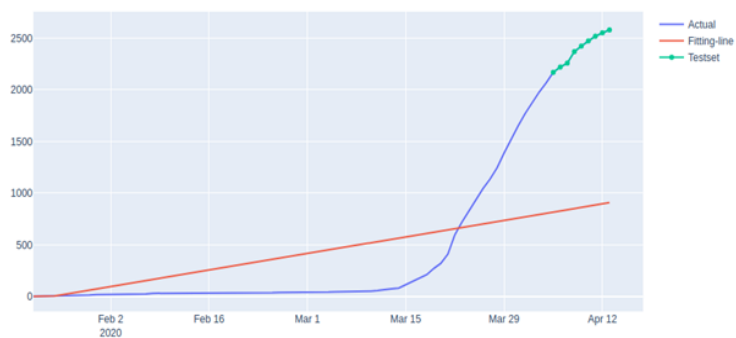
MAE: 351.39625379774304
MSE: 143627.35187444589

Linear-model



MAE: 1337.1801390431528
MSE: 1797198.1742262011

Simple NN



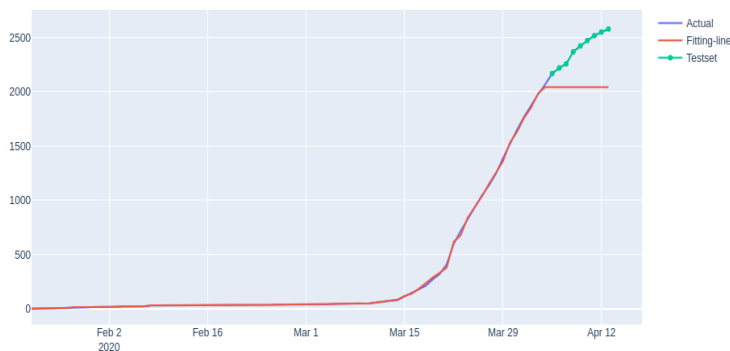
MAE: 1532.8501790364583
MSE: 2362376.746602217

จะเห็นได้ว่า Tree-Based model ให้ค่า Error ต่ำกว่าน้อยที่สุด แต่ไม่สามารถทำนายข้อมูลที่เป็น test ได้เลย เนื่องจาก Tree-Based model เป็นการสร้าง Rule แทนที่จะสร้างความสัมพันธ์ ทำให้พอเจอข้อมูลที่ไม่เคยเจอมาก่อน แล้วไม่ได้มี Rule นั้นจะทำให้ผลลัพธ์แย่ ต่างจากที่ Model แบบอื่นพยายามที่จะสร้างความสัมพันธ์ของตัวแปรต่างๆ แต่ด้วยตัวแปรที่ไม่หลายหลายและน้อยเกินไป ทำให้ผลลัพธ์ยังไม่ดีมากนัก

● Deaths + Recovered

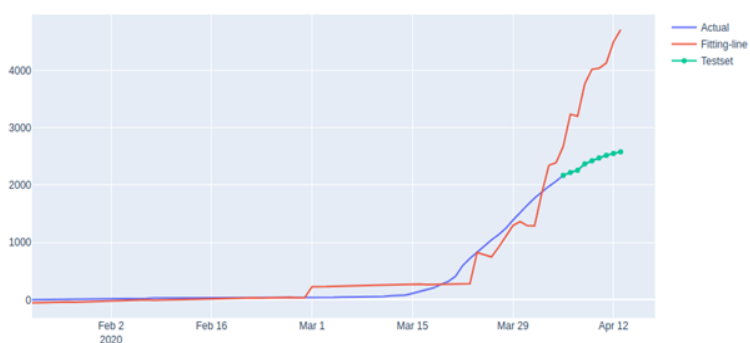
เป็นการนำเอา Timestep มารวมกับยอดการตาย และการรักษาเพิ่มขึ้นมา

Tree-based model



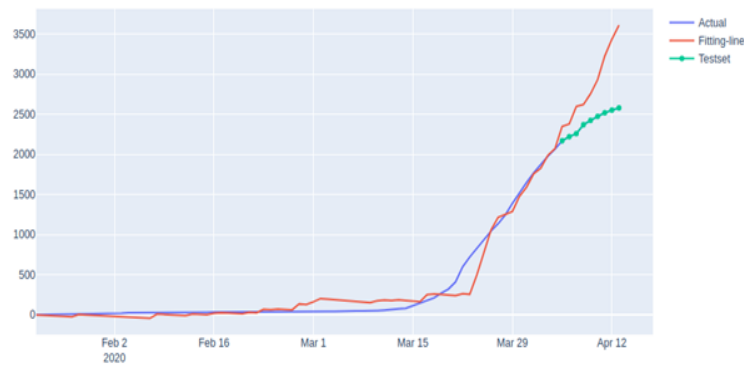
MAE: 351.39625379774304
MSE: 143627.35187444589

Linear-model



MAE: 1409.7490166714028
MSE: 2219637.74807361

Simple NN



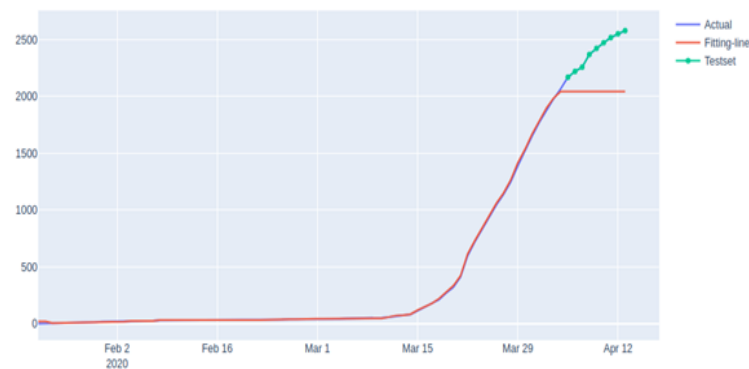
MAE: 482.06049262152777
MSE: 321756.1618236436

ผลลัพธ์แต่ละ Model ทำให้พิสูจน์ให้เราเห็นว่า Tree-Based model ได้ค่าเดิมเลย ดังนั้นเราจะไม่ให้ความสำคัญกับ Tree-Based แล้วด้วยเหตุผลที่เคยกล่าวไปก่อนหน้านี้ และการที่มี Feature มากขึ้นทำให้ model สามารถเรียนรู้ที่ยากขึ้นได้ตาม แต่ผลลัพธ์ก็ยังไม่ดีเท่าที่ควร

● Timestep + Feature Engineer

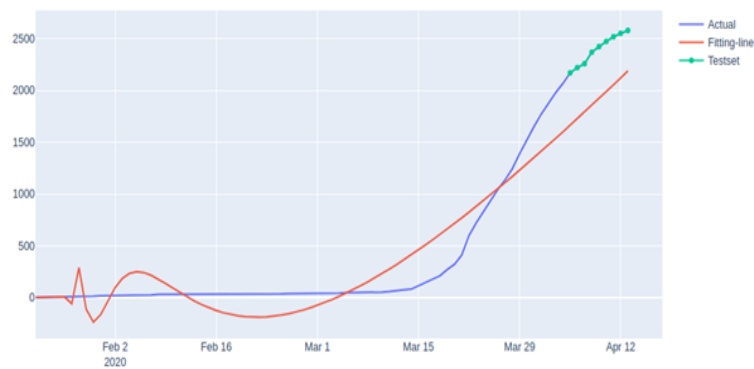
เป็นการนำเอา Timestep มา Aggregate กับ Feature Engineer ที่เรากล่าวไว้ก่อนหน้านี้ทั้งหมด

Tree-based model



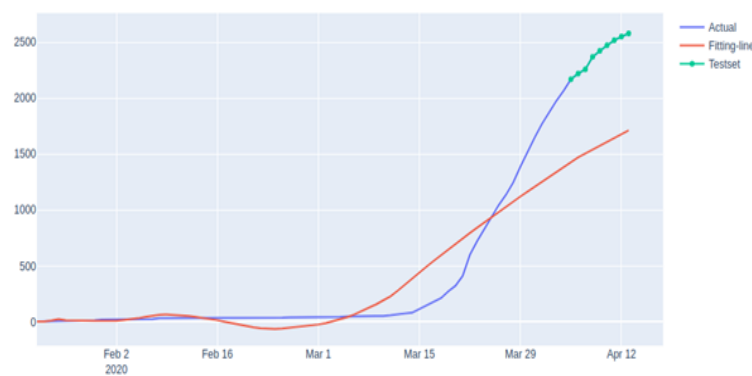
MAE: 351.39625379774304
MSE: 143627.35187444589

Linear-model



MAE: 471.8428740296137
MSE: 224026.88627028847

Simple NN



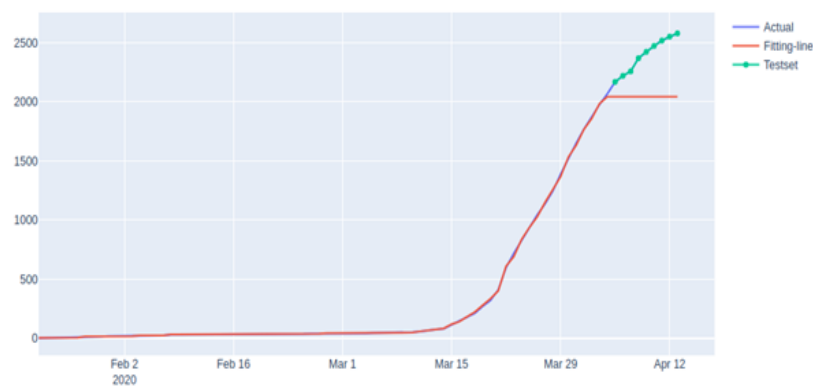
MAE: 821.5978054470486
MSE: 677970.293885587

ผลลัพธ์ดูดีขึ้นถ้าเทียบกับ Timestep อย่างเดียว

● Deaths + Recovered + Feature Engineer

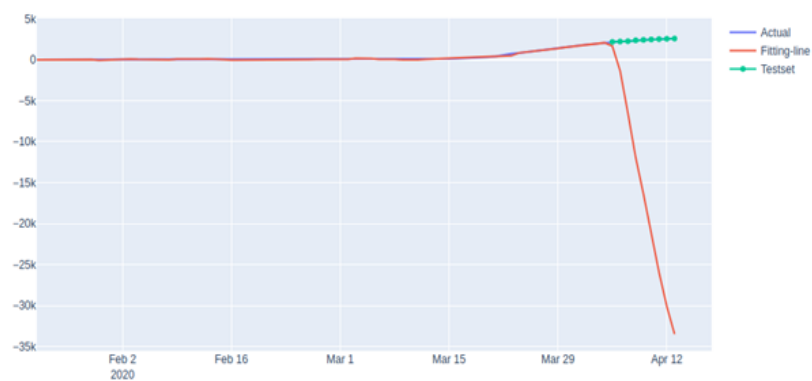
เป็นการนำเอาจำนวนการตาย และการช่วยเหลือได้ มา Aggregate ด้วย Feature Engineering

Tree-based model

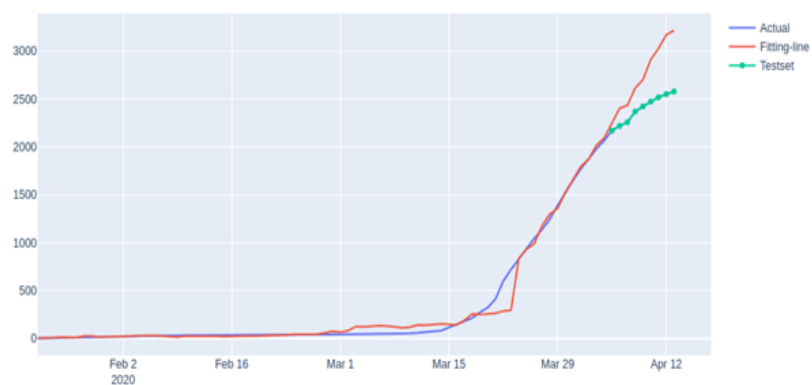


MAE: 351.39625379774304
MSE: 143627.35187444589

Linear-model



Simple NN

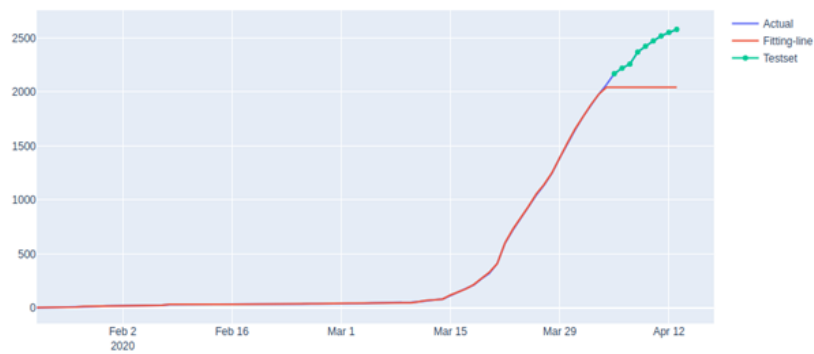


Feature ที่ได้ออกมานั้นมีความหลากหลายและซับซ้อนมากทำให้ส่งผลดีกับ Model ประเภท Neural Network แต่สำหรับ Linear model นั้นแย่ไปเลยอาจจะมาจากการที่มัน Complex เกินไป

- Timestep Polynomial Degree 3

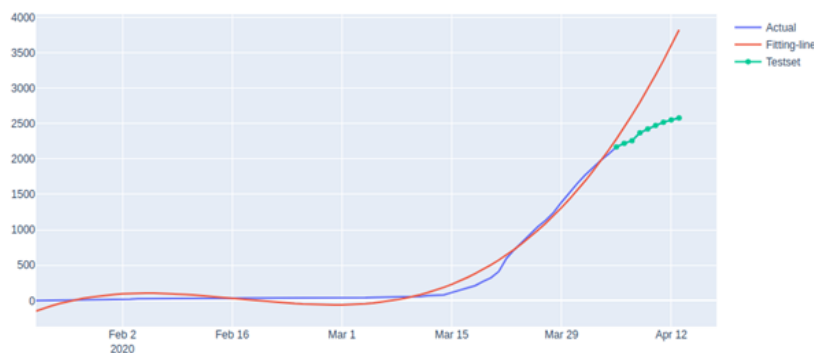
เป็นการนำเอา Timestep มา Aggregate ด้วย Polynomial feature

Tree-based model



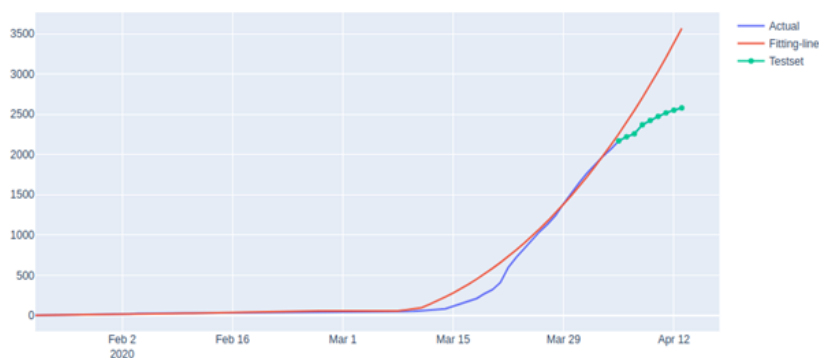
MAE: 351.39625379774304
MSE: 143627.35187444589

Linear-model



MAE: 619.9329186379055
MSE: 513656.26932440576

Simple NN



MAE: 489.47998046875
MSE: 321080.9520906077

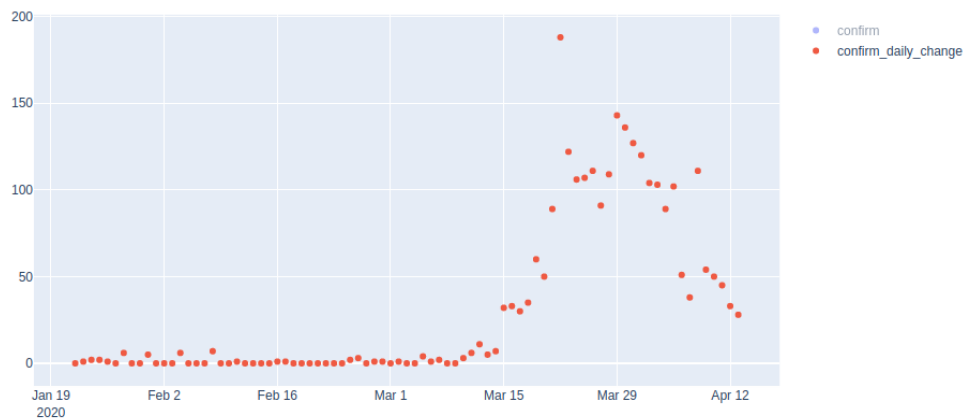
ผลลัพธ์ที่ได้มันดีขึ้นมาก เนื่องจากกราฟไม่ได้มีความเป็น Linear แต่มีความเป็น Polynomial มากกว่า ทำให้เราเห็นว่าการที่ Feature เยอะขึ้นนั้นส่งผลดีกับ Model ที่รองรับความ Complex มากๆได้ แต่มันไม่จำเป็นต้องมี Feature ที่เยอะ ขอแค่มี Feature ที่แสดงถึงความสัมพันธ์จริงๆแค่นั้นก็เพียงพอแล้ว เราจะเห็นได้จากตัวอย่างนี้

● Exponential Model

หลังจากได้ทดลอง Polynomial Regression ไปทำให้เราเห็นว่ามันได้ผลลัพธ์ที่ดีโดยไม่ต้องใช้ Feature ที่เยอะมากมาย และแน่นอนว่ากราฟจริงๆมันไม่ใช่ Polynomial แต่มีความเป็น Exponential มากกว่า เราเลยได้ทำการทดลองโดยอิงจากสมการ และเราเปลี่ยนมาทำนายเคสที่จะเกิดขึ้นในอนาคตแทน

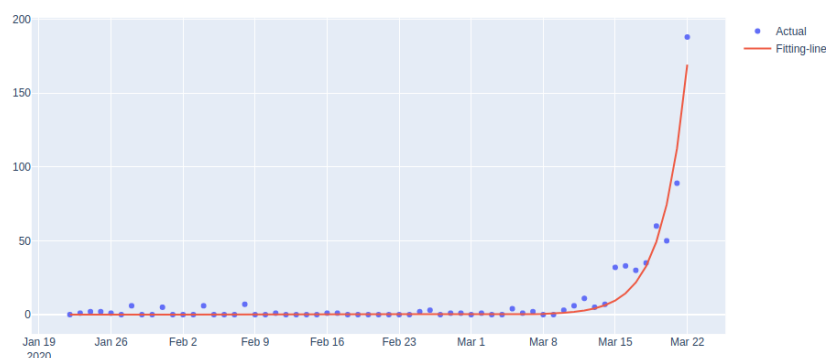
$$f(x) = ae^{b(x-c)}$$

โดย a, b, c คือ parameter ที่เราต้องการทราบ เราได้ทำการสร้าง model จากข้อมูลถึงวันที่ 22 มีนาคม เนื่องจากการเติบโตแบบ Exponential ในช่วงนี้ สาเหตุมาจากประเทศไทยยังไม่ได้ทำการ Curfew แต่พอเรามาดูข้อมูลเต็มๆแล้วนั้น (จนถึงวันที่ 14 เมษายน) ดังรูปต่อไปนี้



และหลังจากการสร้าง model ด้วย Exponential equation จนถึงวันที่ 22 มีนาคม จะได้ผลลัพธ์ดังนี้

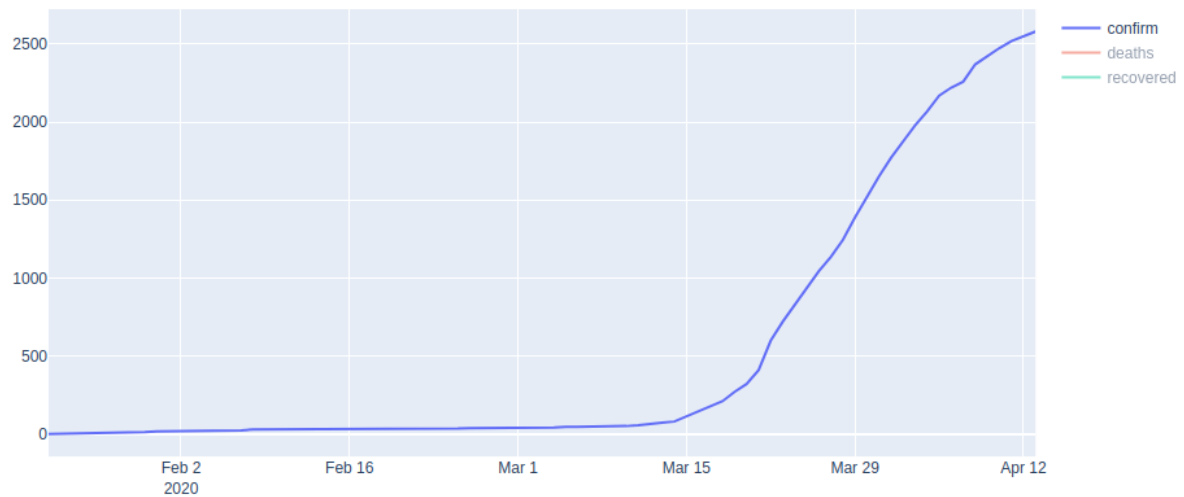
Exponential estimate (2020-01-22 to 2020-03-22)



MAE: 3.161197049503777
MSE: 46.044494914312864

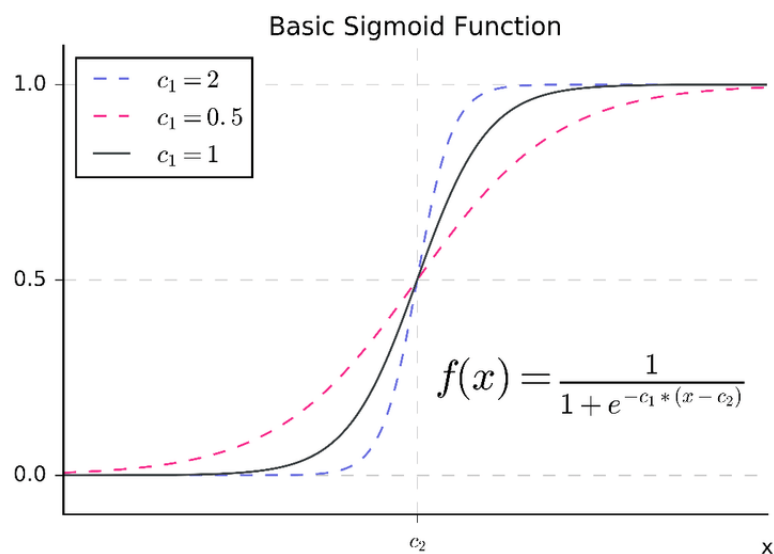
● Sigmoid Model

และถ้าเราสังเกตจากความถี่สะสมของเคสที่เกิดขึ้นตามภาพด้านล่างจะเห็นได้ว่ามันมีการเติบโตที่น้อยลงหรือก็คือเลิกเป็น Exponential แล้วและมีหน้าตาคล้ายกับ Sigmoid มากขึ้น ทำให้เราจะลองทำ Sigmoid function แทนที่จะเป็น Exponential สำหรับจำนวนเคสสะสม



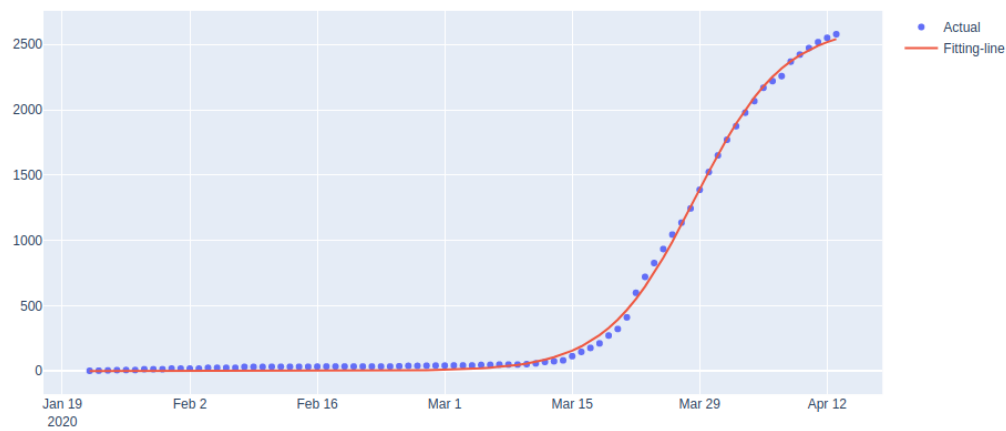
สมการที่เราจะนำมาใช้จะมีการดัดแปลงจากสมการข้างล่างได้สมการดังนี้

$$f(x) = \frac{a}{1 + e^{-b(x-c)}}$$



และได้ทำการสร้าง model ได้ผลลัพธ์ดังนี้

Sigmoid estimate (2020-01-22 to None)



MAE: 28.06544662939251

MSE: 1092.2103923640457

ผลลัพธ์ที่ได้นั้นทำให้เราสร้าง Model ที่สามารถทำนายจำนวนเคสสะสมได้ค่อนข้างแม่นยำ

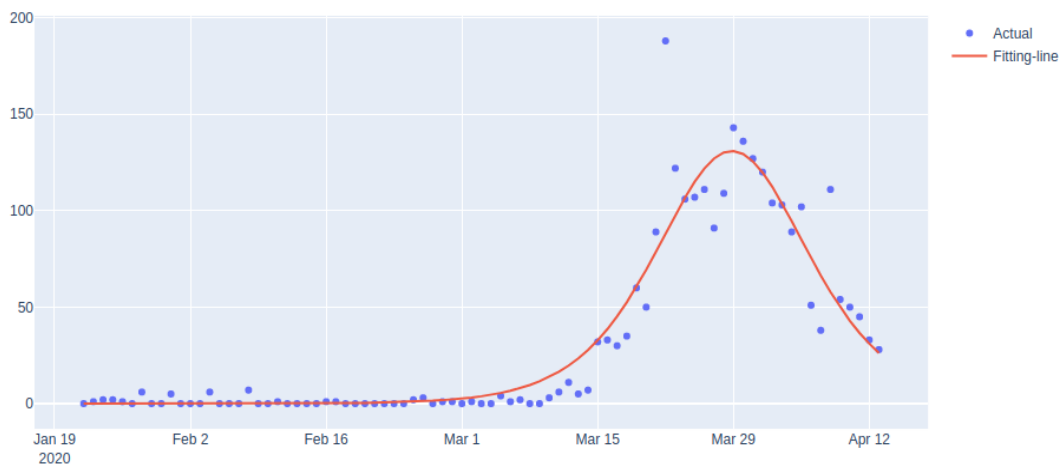
- Logistic Distribution Model

เนื่องจากเราสามารถสร้าง Model ที่สามารถทำนายยอดสะสมต่อไปได้แล้ว เราเลยพยายามที่จะทำนายเคสที่จะเกิดแต่ละวันให้ดีขึ้นจาก Exponential model ที่มีข้อจำกัดอยู่ที่ทำนาย เพียงแค่ช่วงที่เติบโตแบบ Exponential เราได้ไปทำการศึกษาหาความสัมพันธ์ของ Sigmoid function ทำให้เราทราบว่าเราสามารถ map Sigmoid ที่เราสามารถทำนายเคสสะสมให้กลายมา เป็นทำนายจำนวนที่จะเกิดแต่ละวันได้ด้วย Logistic Distribution Function (3)

$$f(x) = \frac{ae^{-b(x-c)}}{1 + e^{-b(x-c)}}$$

ด้วยสมการข้างต้นทำให้เราสร้าง Model ได้ดังต่อไปนี้

Logistic distribution estimate (2020-01-22 to None)



MAE: 7.367697908085666
MSE: 241.92153058208407

ผลลัพธ์ที่ได้นั้นค่อนข้างน่าพึงพอใจ

2.3. สรุปผลการทดลอง

การที่เราใช้ Feature ที่เยอะส่งผลให้ Model มี Feature ที่หลากหลายได้เรียนรู้ และมีผลดีถ้า Model นั้นสามารถรองรับความซับซ้อนสูงๆได้ แต่การที่มี Feature เยอะก็ไม่ได้สำคัญเท่าการมี Feature ที่มีคุณภาพ โดยงานของเราได้ทำการเน้นไปที่การให้ความสำคัญกับการสร้าง Feature ที่ดีที่สุด ซึ่งต้องอาศัยความเข้าใจทางด้านคณิตศาสตร์ และทักษะทางการสังเกต หายที่สุด Model ที่เหมาะสำหรับ ทำนายเคสที่จะเกิดในแต่ละวันก็คือ Logistic Distribution Regression และสำหรับทำนายยอดรวมสะสมก็คือ Sigmoid Distribution

3. Clustering

Credit Card User เป็นข้อมูลของผู้ใช้บัตรเครดิต จำนวน 8950 แต่ละคนมีคุณลักษณะ อธิบายจำนวน 17 Factor เป้าหมายคือแบ่งกลุ่มจากข้อมูล เช่น เอาข้อมูลการใช้จ่ายของลูกค้ามาดูว่าควรเพิ่มวงเงินให้ลูกค้าไหม โดยแต่ละ Factor มีคำอธิบาย ดังนี้

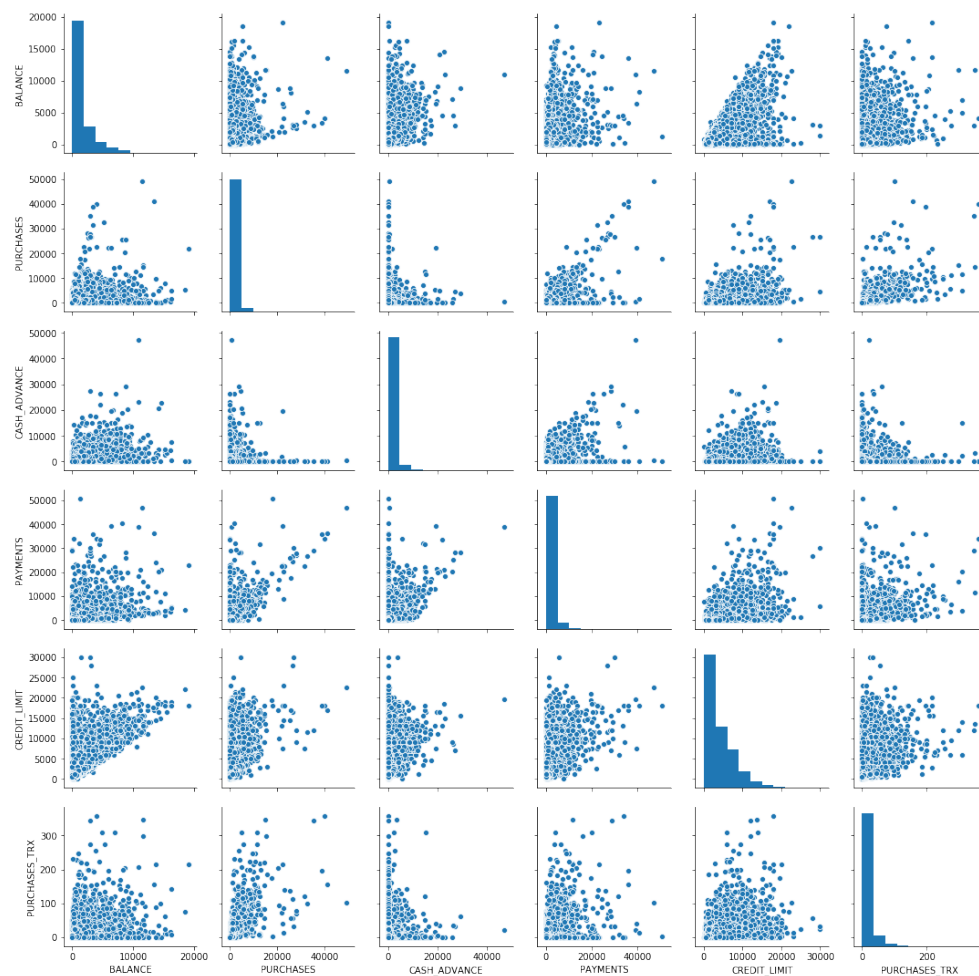
- CUST_ID คือ เลข ID การระบุตัวตนของผู้ถือบัตรเครดิต
- BALANCE คือ ยอดเงินคงเหลือในบัญชีของผู้ถือบัตรเครดิต
- BALANCE_FREQUENCY คือ ยอดคงเหลือมีการอัปเดตบ่อยเพียงใด โดย 0 คือ อัปเดตไม่บ่อยครั้ง และ 1 คือ มีการอัปเดตบ่อยครั้ง
- PURCHASES คือ จำนวนการซื้อจากบัญชี
- ONEOFF_PURCHASES คือ จำนวนการซื้อสูงสุดทำได้ในครั้งเดียว
- INSTALLMENTS_PURCHASES คือ จำนวนการซื้อที่ทำในค่างวดเงินผ่อน
- CASH_ADVANCE คือ เงินสดล่วงหน้าที่ได้รับจากผู้ใช้
- PURCHASES_FREQUENCY คือ ความถี่ในการซื้อสินค้า โดย 0 คือ ไม่ซื้อบ่อย และ 1 คือ ซื้อบ่อย
- ONEOFF_PURCHASES_FREQUENCY คือ ความถี่ในการสั่งซื้อเพียงครั้งเดียว โดย 0 คือ ไม่ซื้อบ่อย และ 1 คือ ซื้อบ่อย
- PURCHASES_INSTALLMENTS_FREQUENCY คือ ความถี่ในการซื้อสินค้าแบบผ่อนชำระ โดย 0 คือ ไม่ซื้อบ่อย และ 1 คือ ซื้อบ่อย
- CASH_ADVANCE_FREQUENCY คือ ความถี่ในการจ่ายเงินสดล่วงหน้า
- CASH_ADVANCE_TRX คือ จำนวนการทำรายการด้วย “เงินสดล่วงหน้า ”
- PURCHASES_TRX คือ จำนวนการทำธุรกรรมการซื้อ
- CREDIT_LIMIT คือ วงเงินบัตรเครดิต
- PAYMENTS คือ จำนวนเงินที่ชำระโดยผู้ใช้

- MINIMUM_PAYMENTS คือ จำนวนเงินขั้นต่ำที่ผู้ใช้จ่าย
- PRC_FULL_PAYMENT คือ เปอร์เซ็นต์ของการชำระเงินเต็มจำนวนโดยผู้ใช้
- TENURE คือ อายุการใช้งานของบริการบัตรเครดิตสำหรับผู้

3.1. Exploration

ในขั้นตอนนี้เราจะเลือกตัวแปรมาทั้งหมด 6 ตัวแปร ได้แก่ BALANCE, PURCHASES, CASH_ADVANCE, PAYMENTS, CREDIT_LIMIT, PURCHASES_TRX และได้ทำการลบข้อมูลที่มีค่า Null ทิ้งทั้งหมด

- ดูว่าข้อมูลมีลักษณะและมีความสัมพันธ์เป็นอย่างไร ด้วยการใช้เทคนิคการนำเสนอข้อมูลด้วยรูปแบบต่างๆ โดยจากรูปด้านล่างนี้เป็นการแสดงโดยที่ยังไม่ได้จัดการกับ Outlier ของข้อมูล



คณะผู้จัดทำได้เลือก 2 วิธีในการดูรูปแบบของข้อมูล คือ Principal Component Analysis (PCA)

และ t-Distributed Stochastic Neighbor Embedding (t-SNE)

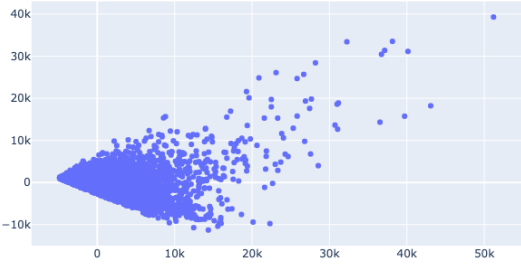
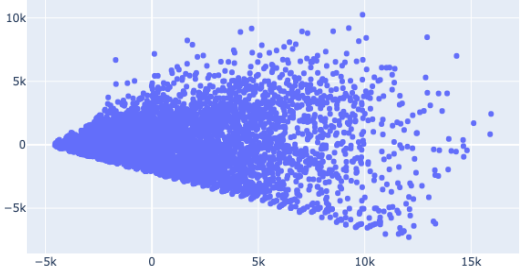
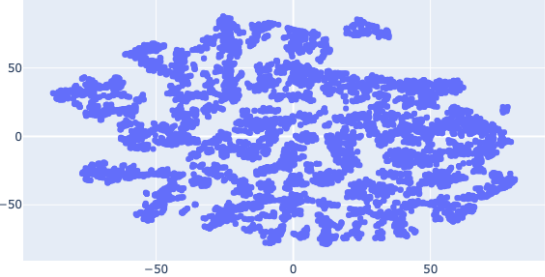
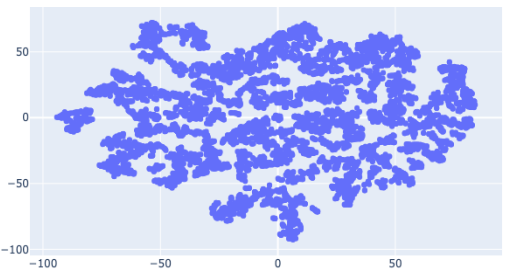
- **Principal Component Analysis (PCA)**

เป็นกระบวนการในการลดมิติของข้อมูลรูปแบบหนึ่ง ด้วยการตัด features ที่ไม่สำคัญออก โดยการ Normalize data ด้วย Standard deviation ต่อด้วยการทำ Covariance matrix จากนั้นทำตามทฤษฎีของ Eigen ผลลัพธ์ที่ได้คือค่าที่ใช้ในการสกัดข้อมูล หลังจากนั้นเราจะสามารถสกัด Feature ที่สมควรแก่การนำไปใช้ในการวิเคราะห์

- **t-Distributed Stochastic Neighbor Embedding (t-SNE)**

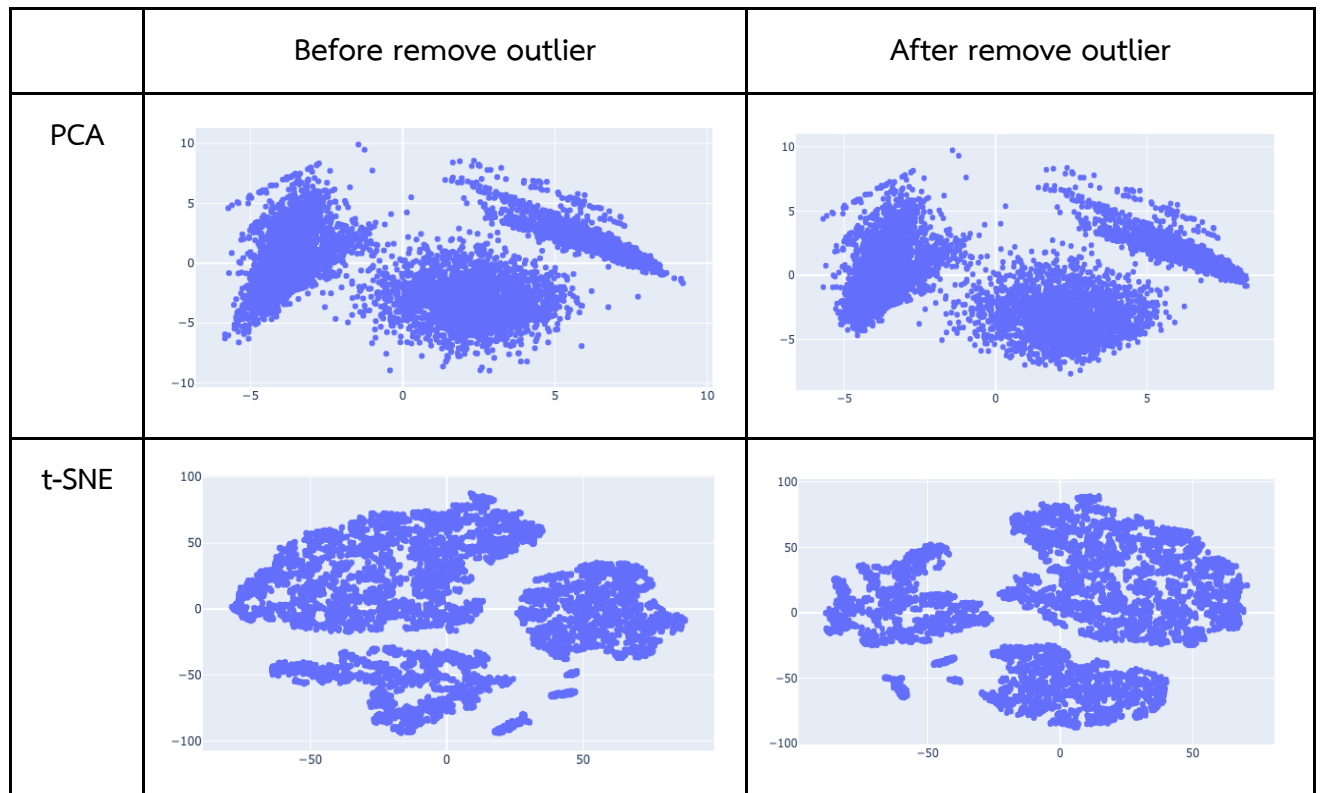
เป็นกระบวนการในการลดมิติของข้อมูลเช่นกัน แต่เป็นการลดมิติแบบไม่เชิงเส้น ซึ่งตรงกันข้ามกับ PCA โดย t-SNE จะทำการคำนวณหา Probability of similarity และพยายามเรียนรู้เป็น Iteration โดยทำให้ตัวที่มี Embedding คล้ายกัน ให้มาอยู่ใกล้กันและจัดเป็นกลุ่มเดียวกัน (4) (5)

● Original - Features

	Before remove outlier	After remove outlier
PCA	 A PCA scatter plot showing a dense cluster of blue points. The x-axis ranges from 0 to 50k with major ticks every 10k. The y-axis ranges from -10k to 40k with major ticks every 10k. A single outlier point is visible at approximately (50k, 40k).	 A PCA scatter plot showing the same data after removing the outlier. The x-axis ranges from -5k to 15k with major ticks every 5k. The y-axis ranges from -5k to 10k with major ticks every 5k. The cluster is more compact and centered around the origin.
t-SNE	 A t-SNE scatter plot showing a dense, irregular cluster of blue points. The x-axis ranges from -50 to 50 with major ticks at -50, 0, and 50. The y-axis ranges from -50 to 50 with major ticks at -50, 0, and 50.	 A t-SNE scatter plot showing the same data after removing the outlier. The x-axis ranges from -100 to 50 with major ticks at -100, -50, 0, and 50. The y-axis ranges from -100 to 50 with major ticks at -100, -50, 0, and 50. The cluster appears slightly more compact than in the 'before' plot.

- Logarithm - transformation

Log - transformation เป็นวิธีการหนึ่งในการทำ Feature Engineering เพื่อให้มีการกระจายตัวของข้อมูลกว้างมากขึ้น ส่งผลให้ข้อมูลมีการกระจายตัวออกจากกันเพิ่มมากขึ้น โดยเราจะแสดงเปรียบเทียบให้เห็นทั้งสองอย่าง (6)



3.2. Data preparation

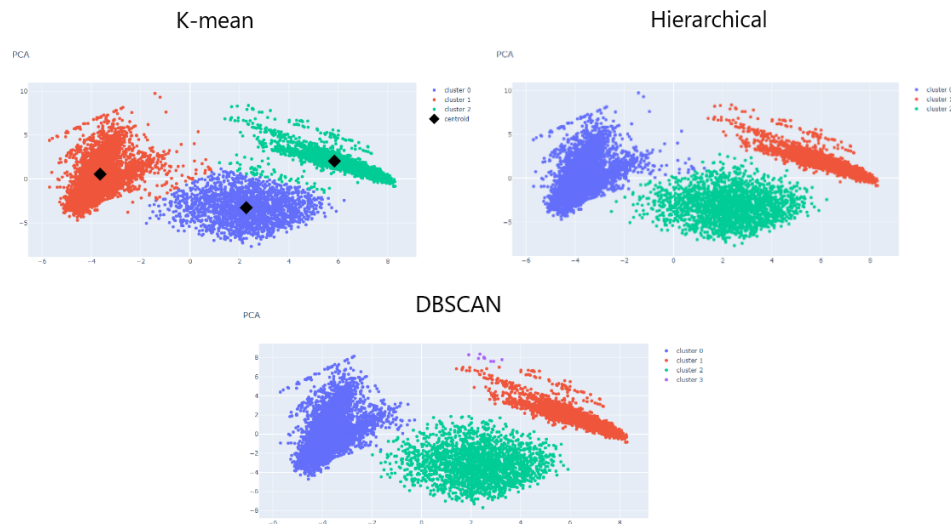
เนื่องจากข้อมูลมี Outlier ทางคณะผู้จัดทำจึงเลือกที่จะจัดการกับ Outlier โดยใช้

Standard Deviation

```
def StandardDeviation(df, coef_sd=4) :  
    """  
    Remove outlier with Standard Deviation.  
    """  
  
    df_tmp = df.copy()  
  
    mean_df = df.mean()  
    std_df = df.std()  
  
    # inside distribution  
    in_range = df_tmp[(df_tmp >= mean_df - coef_sd * std_df) & \  
                      (df_tmp <= mean_df + coef_sd * std_df)]  
  
    # outside distribution  
    out_of_range = df_tmp[(df_tmp < mean_df - coef_sd * std_df) & \  
                          (df_tmp > mean_df + coef_sd * std_df)]  
  
    return in_range, out_of_range  
  
for feat in features:  
    df[feat] = StandardDeviation(df[feat])[0]  
  
df = df[features].dropna()  
  
df.to_csv('./data/01_clean.csv')
```

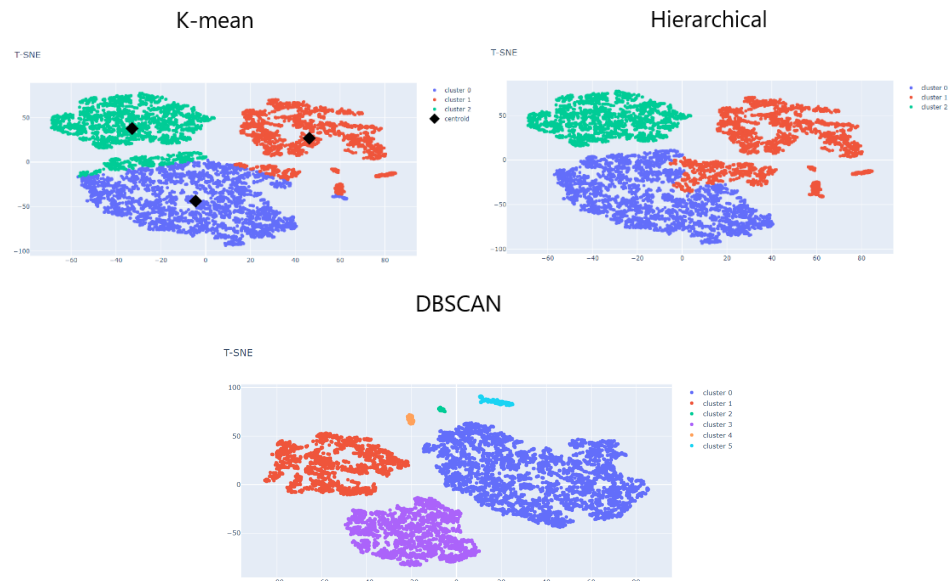
3.3. Modeling

ทางผู้จัดทำได้เลือกโมเดลมาทั้งหมด 3 แบบ โดยทดสอบจาก 2 Feature ได้แก่ PCA, t-SNE



ภาพที่ใช้ Feature PCA

จากภาพ โมเดล K-mean กลุ่มสีแดงและกลุ่มสีเขียว มีการ cluster ผิดกลุ่ม ทำให้มีบางตัวไปปะปนอยู่ในกลุ่มสีน้ำเงิน ซึ่งเป็นเพราะว่าตัวสีแดงที่ควรอยู่ในกลุ่มสีน้ำเงินนั้น มี Distance ใกล้กับ Centroid ของกลุ่มสีแดงมากกว่า เลยทำให้เห็นว่า การใช้ Distance ในการแบ่งกลุ่มอย่างเดียว ยังไม่ดีพอ ควรจะมี Density มาเกี่ยวข้องด้วย อย่างในตัวอย่างที่ใช้โมเดล DBSCAN มีการแบ่งกลุ่มได้ดีที่สุด แต่จะ Sensitive กับ Outlier ทำให้เห็นได้ว่ามีกลุ่มเพิ่มขึ้นมา 1 กลุ่ม



ภาพที่ใช้ Feature t-SNE

จากภาพ โมเดล K-mean ยังมีส่วนของกลุ่มสีเขียวและแดงมาปะปนอยู่ในกลุ่มสีน้ำเงิน โมเดล Hierarchical มีส่วนของกลุ่มแดงเข้ามาปะปนกันเยอะพอสมควร โมเดล DBSCAN ยังมีจำนวนกลุ่มที่มากกว่าอีก 2 model ซึ่งกลุ่มที่เกินมาของ DBSCAN อาจมองเป็น Outlier ดังนั้นผู้จัดทำเลือก DBSCAN มีความเหมาะสมผลมากที่สุด เพราะในความเป็นจริงเราไม่รู้ว่ามีข้อมูลมีกี่กลุ่ม ทำให้ไม่ต้องกำหนดค่า K เลยทำให้เป็นข้อได้เปรียบ

3.4. สรุปผลการทดลอง

เนื่องจากในตอนแรกชุดข้อมูลก็ยังคงค่อนข้างจัดกลุ่มยาก จึงใช้ Log – Transformation ในการจัดการข้อมูล แล้วทำการทดลองโมเดลทั้งหมด 3 แบบ ได้แก่ K - Mean, Hierarchical, DBSCAN ซึ่งทั้ง 3 โมเดล ก็ให้ผลลัพธ์ที่มีความแตกต่างกันเพียงเล็กน้อย แต่ความต่างระหว่าง PCA กับ t-SNE นั้นมี ความแตกต่างกันค่อนข้างมาก ซึ่งเราต้องเลือกใช้โมเดลให้เหมาะสมกับ feature ด้วย ซึ่งผลลัพธ์หลังจากทำ Log – Transformation แล้วมาใช้ feature PCA, t-SNE ทำให้ข้อมูลแบ่งกลุ่มได้อย่างชัดเจนแทบจะดูด้วยตา ก็บอกได้ สุดท้ายนี้ทางผู้จัดทำเลือก โมเดล DBSCAN และใช้ Feature PCA ในการ Clustering เพราะเป็นการที่แบ่งกลุ่มได้อย่างชัดเจนที่สุด ส่วนกลุ่มที่เป็นส่วนน้อยและเกินออกมาอาจเป็น Outlier ของข้อมูล

References

1. -Proposed TDSP data science process The two figures above, figure 5 and... | Download Scientific Diagram [Internet]. [cited 2020 May 3]. Available from: https://www.researchgate.net/figure/Proposed-TDSP-data-science-process-The-two-figures-above-figure-5-and-figure-6-which_fig5_336530802
2. Frazier PI. A Tutorial on Bayesian Optimization. 2018 Jul 8 [cited 2020 May 3]; Available from: <http://arxiv.org/abs/1807.02811>
3. Logistic distribution - Wikipedia [Internet]. [cited 2020 May 3]. Available from: https://en.wikipedia.org/wiki/Logistic_distribution
4. Wattenberg M, Viégas F, Johnson I. How to Use t-SNE Effectively. Distill [Internet]. 2016 Oct 13 [cited 2020 May 3];1(10):e2. Available from: <http://distill.pub/2016/misread-tsne>
5. การลดมิติข้อมูลโดยใช้ PCA และ t-SNE - Konakona Chan - Medium [Internet]. [cited 2020 May 3]. Available from: <https://medium.com/@konakona7393/การลดมิติข้อมูลโดยใช้-pca-และ-t-sne-e6679f44143d>
6. Feature engineer: การแปลงข้อมูลโดยใช้ค่าล็อก (Log Transformation) | Big Data Experience Center [Internet]. [cited 2020 May 3]. Available from: <http://bigdataexperience.org/feature-engineer-with-log-transformation/>