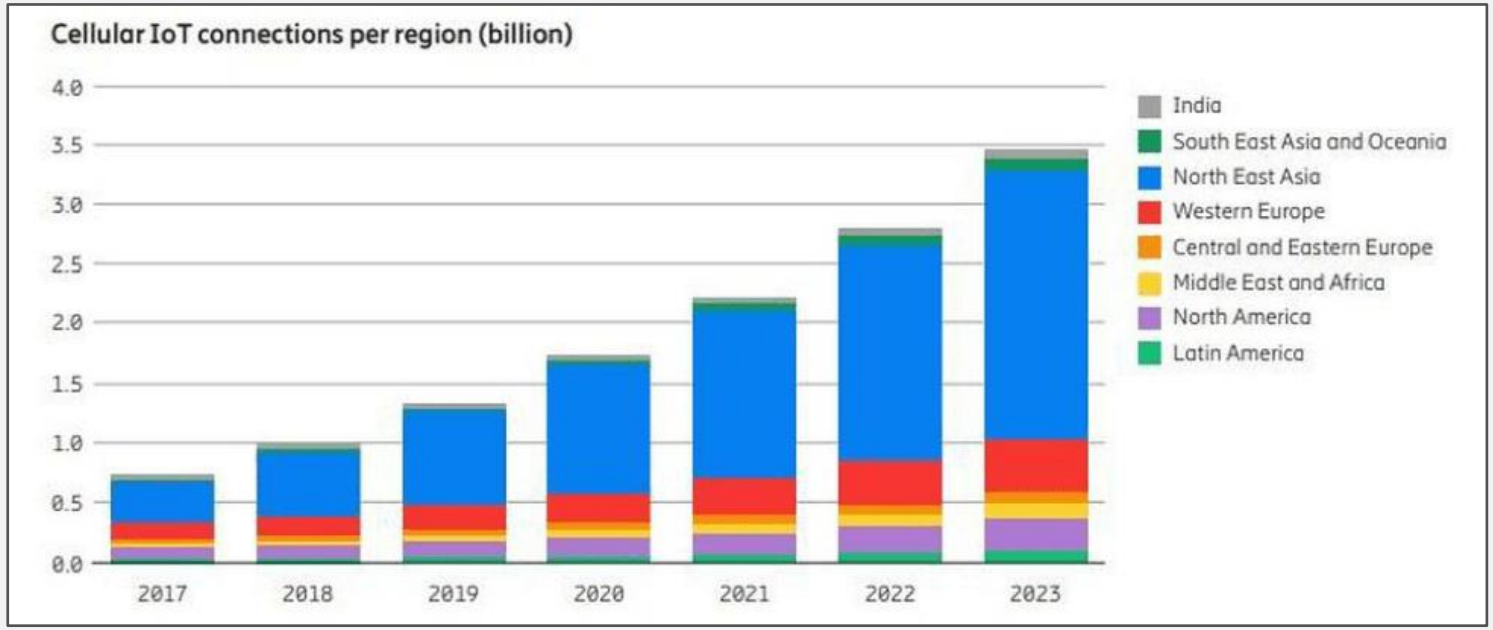
Regression For Machine learning

Motivation

Catch up the technical world



Rapidly growing IoT industry
According to IDC, the worldwide technology spending on the Internet Of Things would reach $1 trillion in 2022 at the CAGR of 13.6% over the period of 2017-2022.
According to the prediction of Ericsson, the IoT of the cellular industry is expected to reach $3.5 billion in 2023 at a CAGR of 30%.

From :Clicdata.com

https://www.clicdata.com/blog/top-14-crucial-data-science-trends-in-2019/

Top skills for Data Scientists

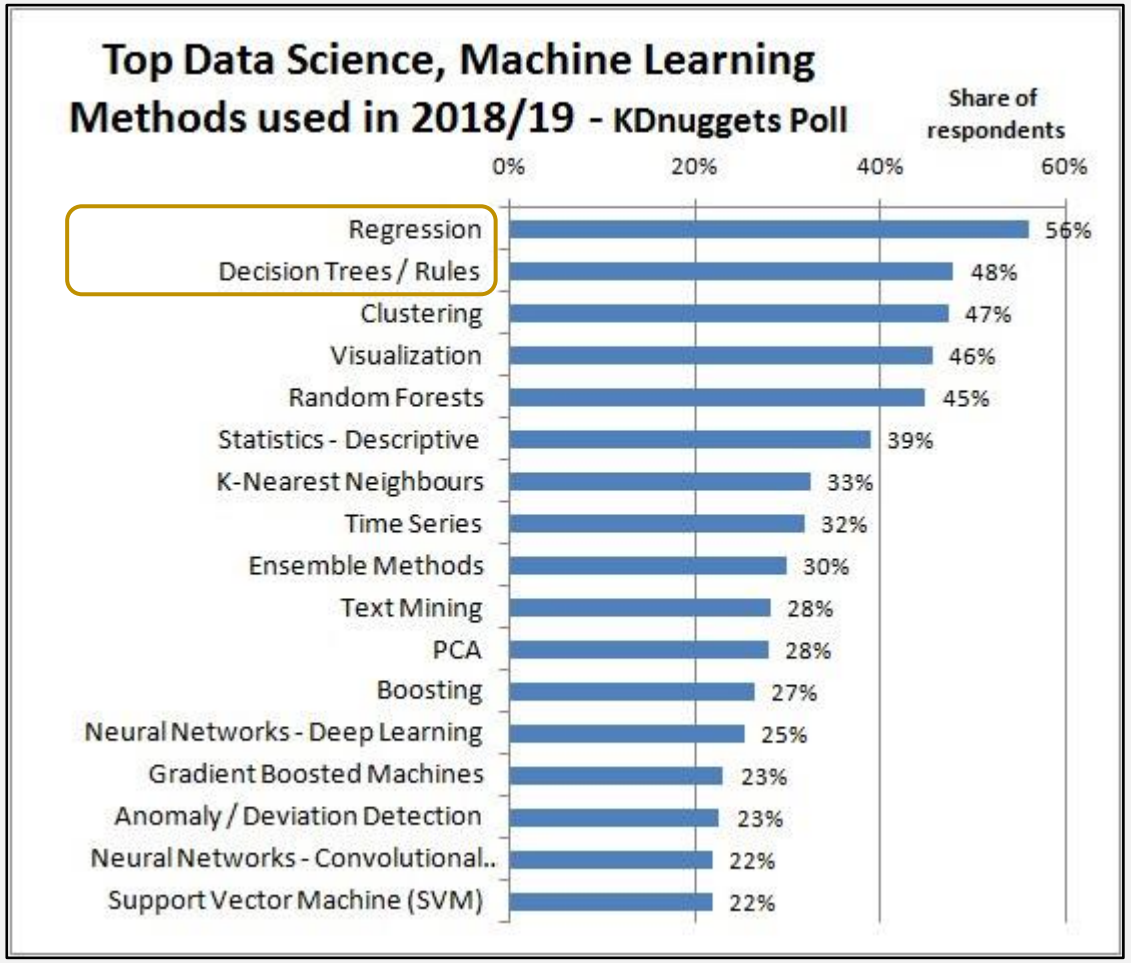
Top skills: Data Scientist

Sept 2018 to Sept 2019

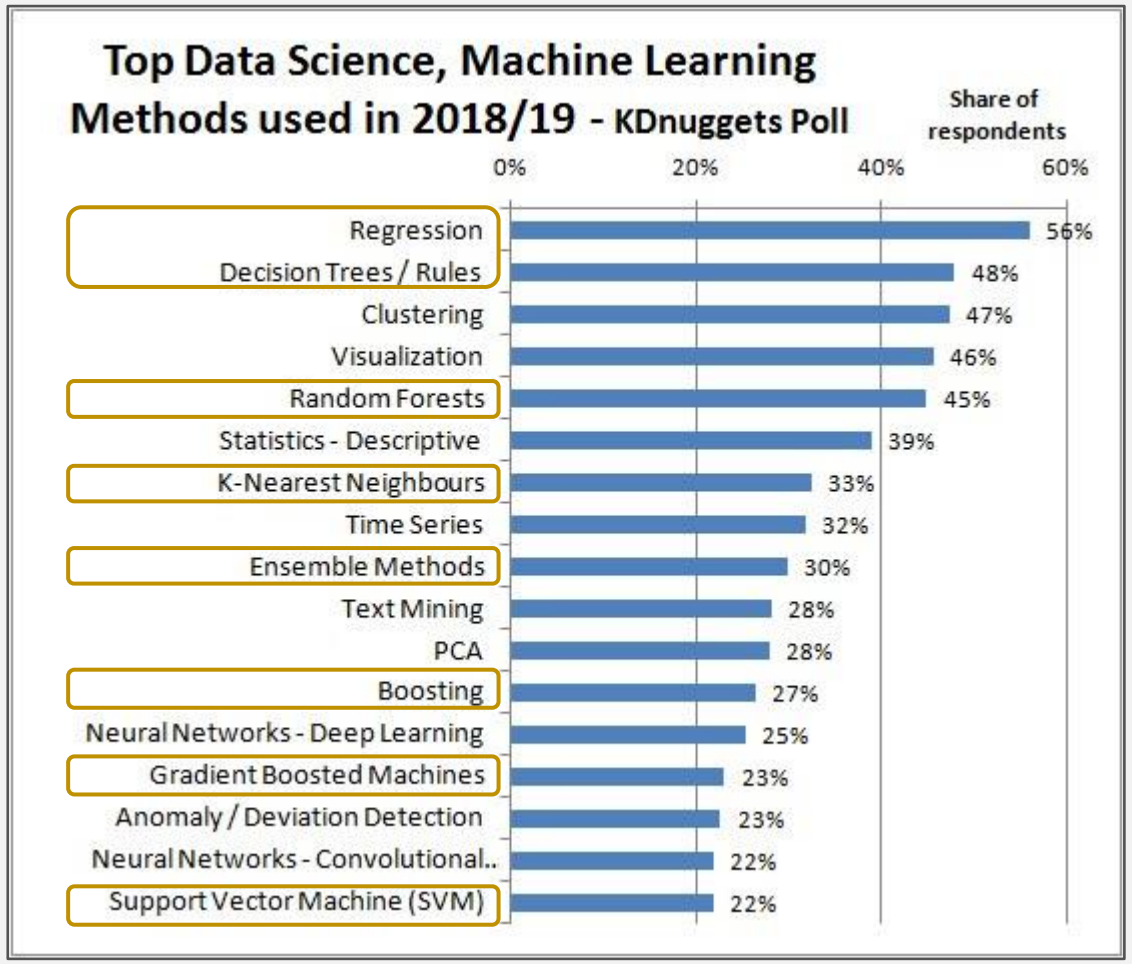
Rank	Skill	Percent of jobs
1	python	79%
2	machine learning	72%
3	r	64%
4	sql	53%
5	hadoop	29%
6	spark	28%
7	java	25%
8	sas	24%
9	tableau	21%
10	deep learning	20%

Source: Indeed

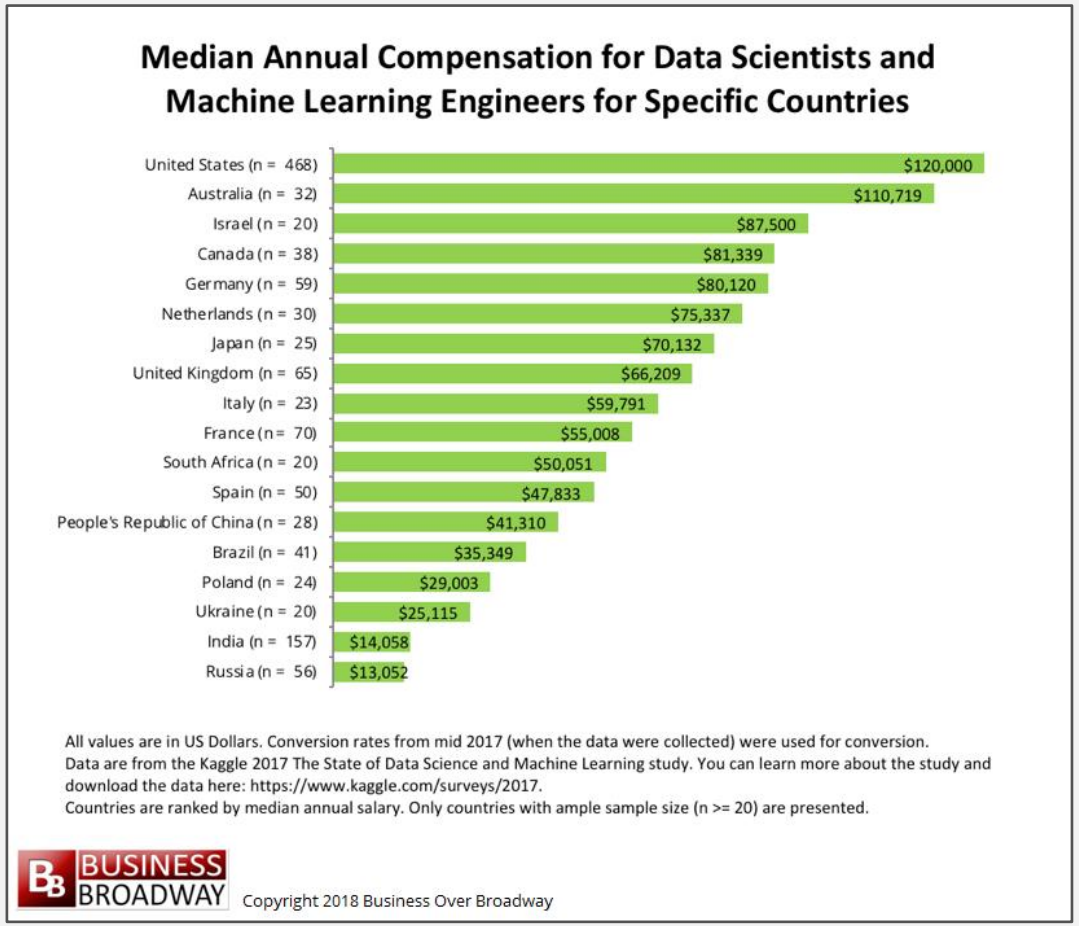
indeed

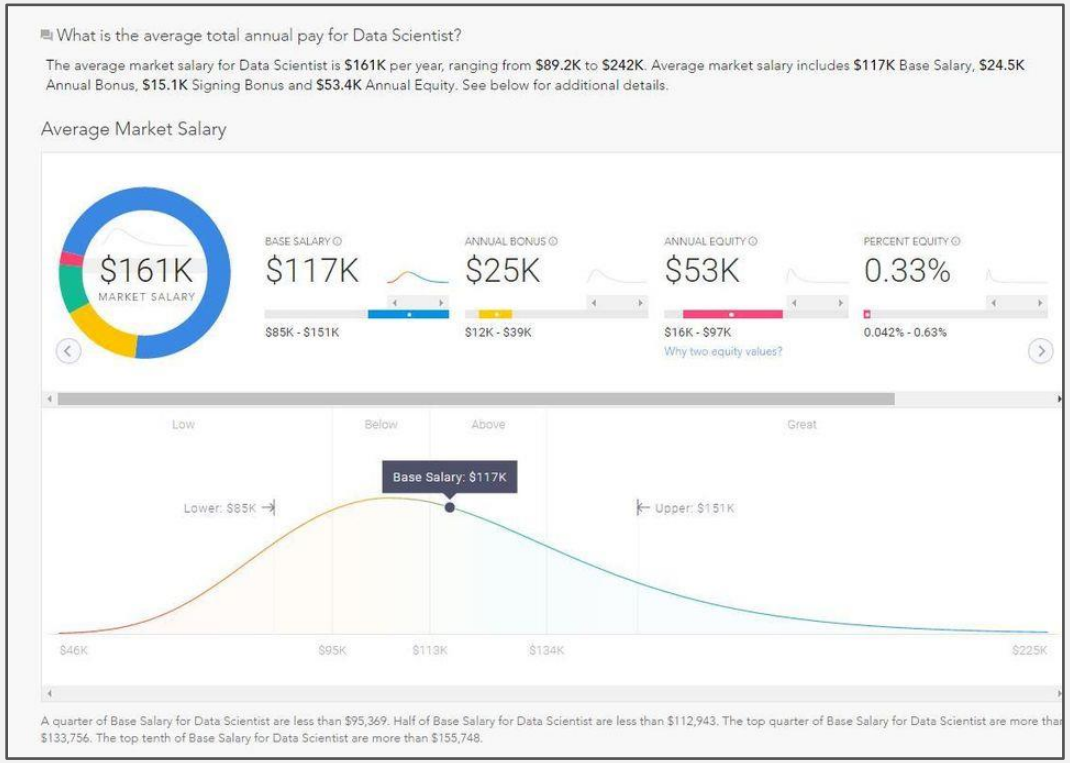


Top skills for Data Scientists

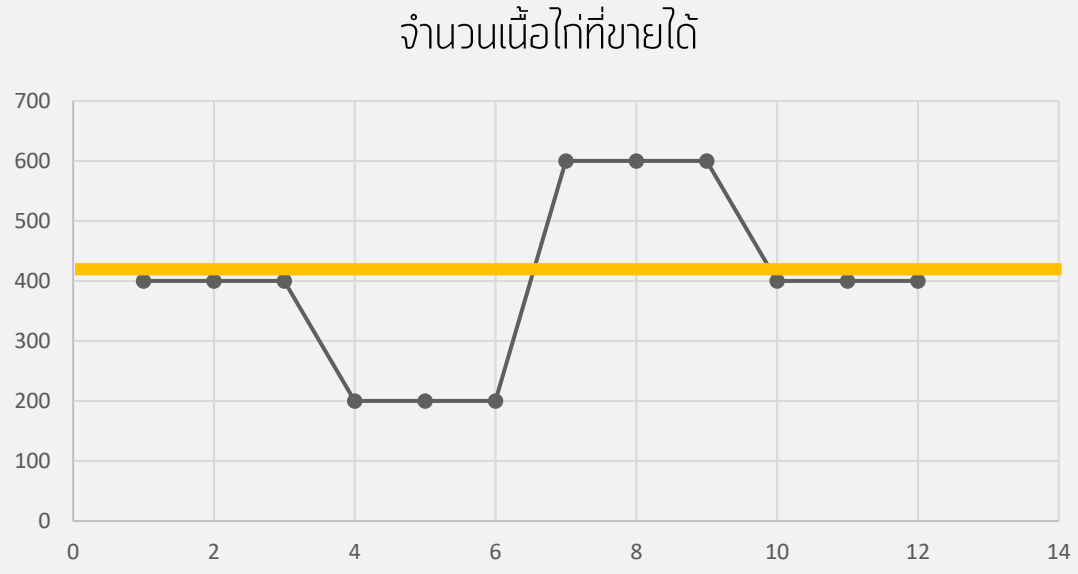


Salary


Median Annual Compensation for Data Scientists and Machine Learning Engineers for Specific Countries
United States (n = 468) $120,000
Australia (n = 32) $110,719
Israel (n = 20) $87,500
Canada (n = 38) $81,339
Germany (n = 59) $80,120
Netherlands (n = 30) $75,337
Japan (n = 25) $70,132
United Kingdom (n = 65) $66,209
Italy (n = 23) $59,791
France (n = 70) $55,008
South Africa (n = 20) $50,051
Spain (n = 50) $47,833
People's Republic of China (n = 28) $41,310
Brazil (n = 41) $35,349
Poland (n = 24) $29,003
Ukraine (n = 20) $25,115
India (n = 157) $14,058
Russia (n = 56) $13,052
All values are in US Dollars. Conversion rates from mid 2017 (when the data were collected) were used for conversion.
Data are from the Kaggle 2017 The State of Data Science and Machine Learning study. You can learn more about the study and download the data here: https://www.kaggle.com/surveys/2017.
Countries are ranked by median annual salary. Only countries with ample sample size (n >= 20) are presented.
BUSINESS BROADWAY
Copyright 2018 Business Over Broadway


What is the average total annual pay for Data Scientist?
The average market salary for Data Scientist is $161K per year, ranging from $89.2K to $242K. Average market salary includes $117K Base Salary, $24.5K Annual Bonus, $15.1K Signing Bonus and $53.4K Annual Equity. See below for additional details.
Average Market Salary
$161K
MARKET SALARY
BASE SALARY
$117K
$85K - $151K
ANNUAL BONUS
$25K
$12K - $39K
ANNUAL EQUITY
$53K
$16K - $97K
Why two equity values?
PERCENT EQUITY
0.33%
0.042% - 0.63%
Low
Below
Above
Great
Base Salary: $117K
Lower: $85K
Upper: $151K
$46K
$95K
$113K
$134K
$225K
A quarter of Base Salary for Data Scientist are less than $95,369. Half of Base Salary for Data Scientist are less than $112,943. The top quarter of Base Salary for Data Scientist are more than $133,756. The top tenth of Base Salary for Data Scientist are more than $155,748.

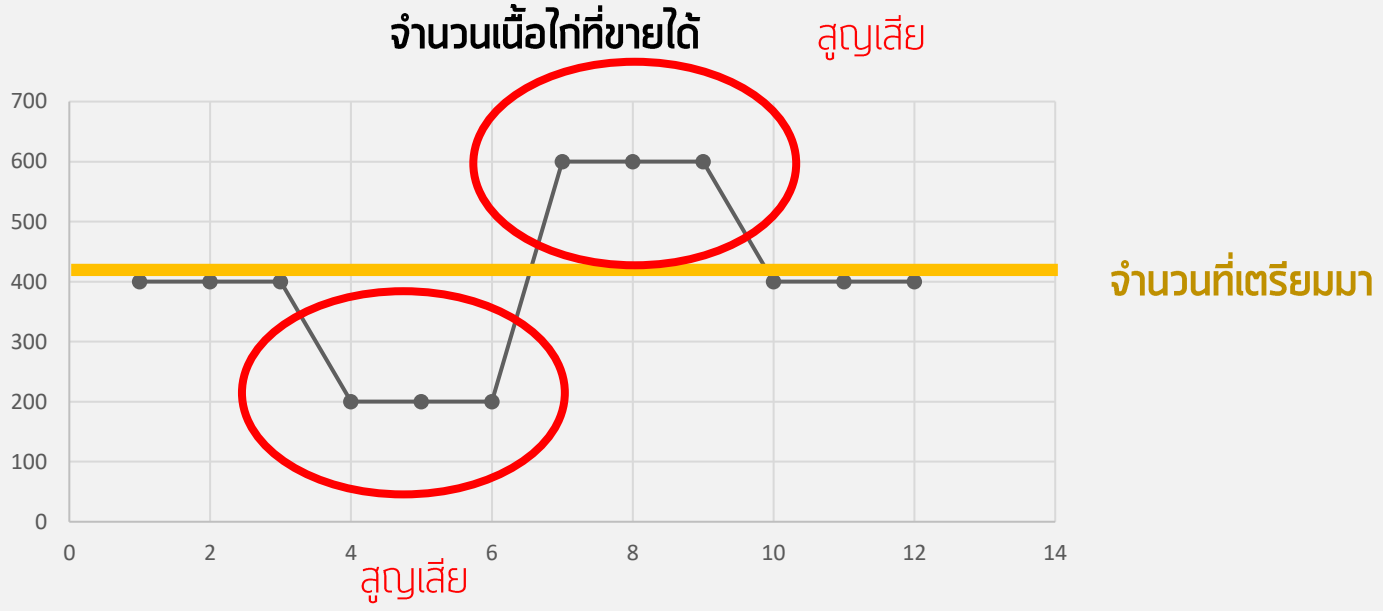
Business application with data



จำนวนที่เตรียมมา

หากพ่อค้าเตรียมเนื้อไก่มาเท่ากันทุกวัน

Easy Business application with data



Course Outline

Course Outline

Principle

1. **General Introduction :**
   - What is machine learning
   - What is regression
   - Type of regression
   - Linear regression theory

2. **Simple Linear Regression**
   - Simple linear regression theory
   - Term definition
   - Regression Cost
   - Coefficient Prove

3. **General Machine learning Methodology**
   - Data Split
   - Prediction
   - Model Evaluation
   - Optimization
   - Parameter Tuning

4. **First R session**
   - R session
   - Interpreting Result

5. **Multiple Linear Regression**
   - Multiple Linear Regression Theory
   - Interpreting Result For Multiple Linear Regression
   - Non-Linear regression
   - Multiple Linear Regression Coefficient Prove
   - R session Multiple regression

Variation

6. **Potential Problem of Complex Modeling:**
   - Underfit vs Overfit
   - Ridge Regression
   - Lasso Regression
   - Feature Scaling
   - R session Regularized regression

7. **Tree Regression**
   - Decision Tree Regression
   - Random Forest Regression
   - Gradient Boosting Regression
   - R session Tree regression

8. **Support Vector Machine Regression**
   - Linear SVM regression
   - Kernel Trick
   - SVM Polynomial kernel Regression
   - SVM Gaussian kernel Regression
   - R session SVM regression

9. **Distance Regression**
   - K-Nearest Neighbor Regression
   - R session Distance regression

10. **Ensemble Regression**
    - Ensemble regression idea
    - CaretEnsemble
    - R session Ensemble regression

Wrap up

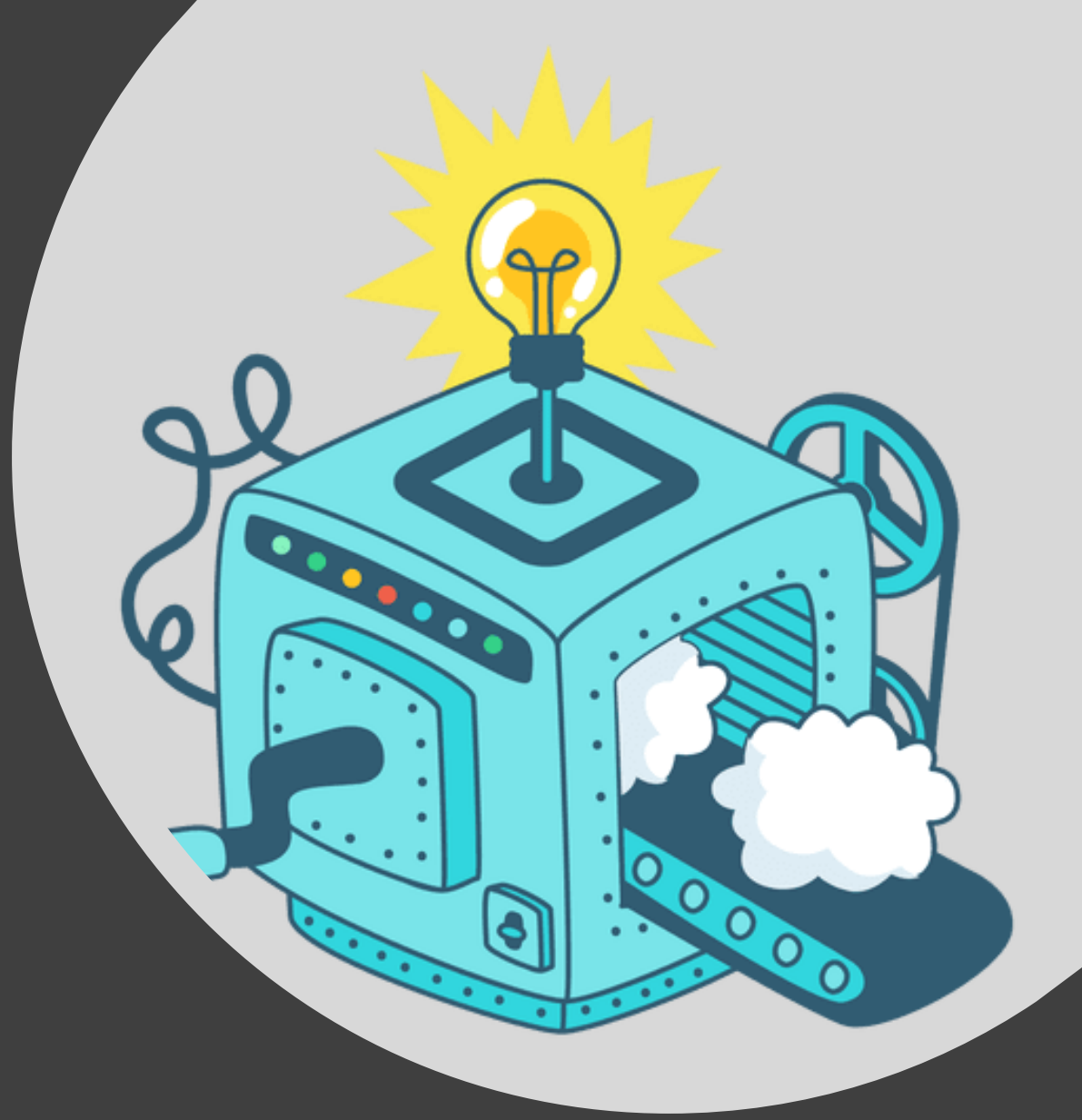
11. **Regression Preparation:**
    - Ideally Dataset
    - Descriptive Statistic
    - Missing Value Exploration
    - Correlation Exploration
    - Split Dataset
    - Scaling Dataset
    - Additional Preprocessing
    - Training the Model
    - Interpreting Result
    - Prediction
    - Evaluation

General Introduction

What is Machine learning?

- การทำให้ Computer เรียนรู้ด้วยตนเองโดยใช้ข้อมูล
- การเรียนรู้ จากข้อมูลนั้นจะออกมาในรูปแบบของ Model

- ประเภทของ Machine learning จะมี
  - **Supervised Learning**
  - Unsupervised Learning
  - Reinforcement Learning



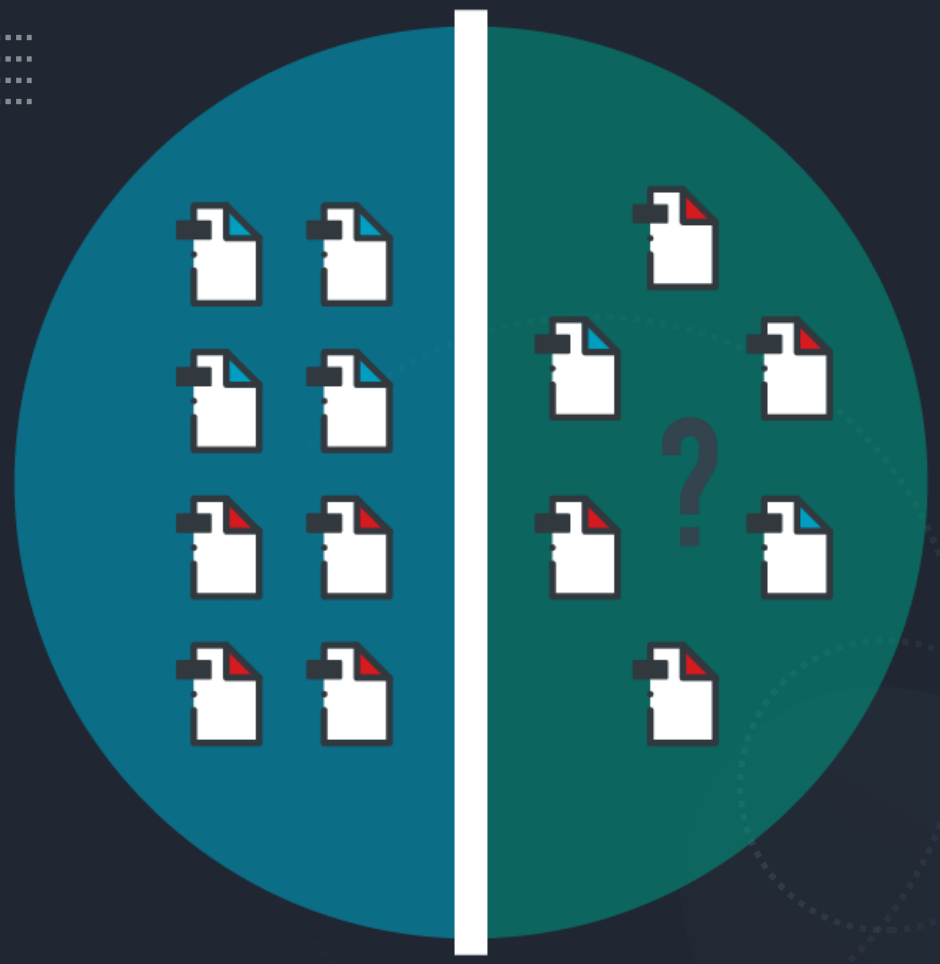
Supervised Machine learning

ทำ model เพื่อไป predict ข้อมูลที่ไม่เคยเห็นมาก่อน

2 ประเภท

- 1. categorical data (ข้อมูล output เป็นกลุ่ม category เช่น คำตอบ yes/no จะมี 2 cate คือ yes หรือ no) -> classification

- 2. numerical data (ข้อมูล output เป็นตัวเลข หรือ numerical data เช่น ยอดขายเนื้อไก่ กี่ kg) -> **Regression**



We will first use linear regression!!

Regression Modeling Objective

Regression Modeling มีเป้าหมายหลัก 2 อย่าง

- 1. ทำนายข้อมูล (quantitative / ตัวเลข) ที่เราไม่เคยเห็นมาก่อน เช่น

1.1 ทำนายยอดขายของ เพื่อที่บริษัทจะได้จัดการ Inventory ถูก

1.2 ทำนายราคาวัตถุดิบ เพื่อที่จะประเมิน ต้นทุนเบื้องต้นได้

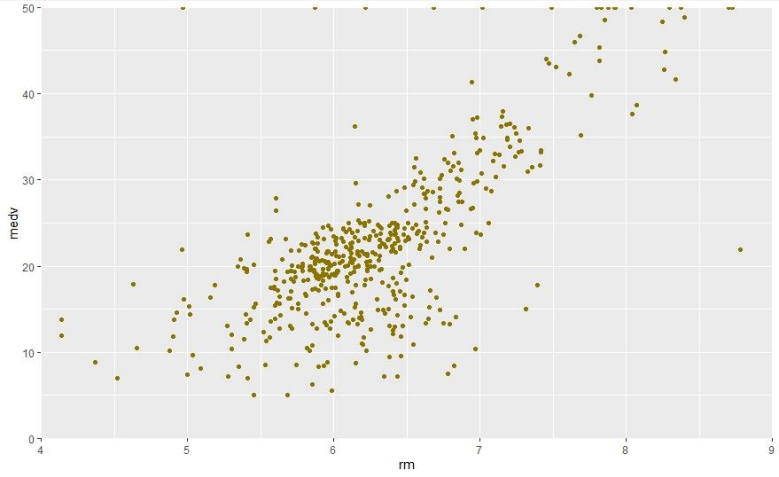
- 2. อธิบายความสัมพันธ์ทางธุรกิจ

2.1 ดูว่าระหว่าง ปริมาณฝุ่น มีผลต่อยอดขายของบริษัทไหม

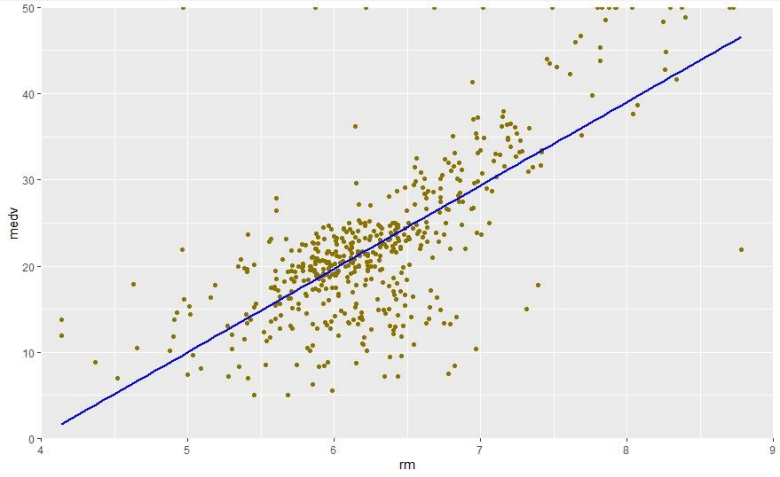
2.2 ดูว่าระหว่าง ราคายางโลก มีผลต่อ ราคาวัตถุดิบที่เราต้องการไหม

Linear Regression Model (Fundamental)

- สร้าง **<u>เส้นตรง</u>** ที่ fit ข้อมูลได้ดีที่สุด
- นำไป predict ข้อมูลที่ไม่เคยเห็นมาก่อน



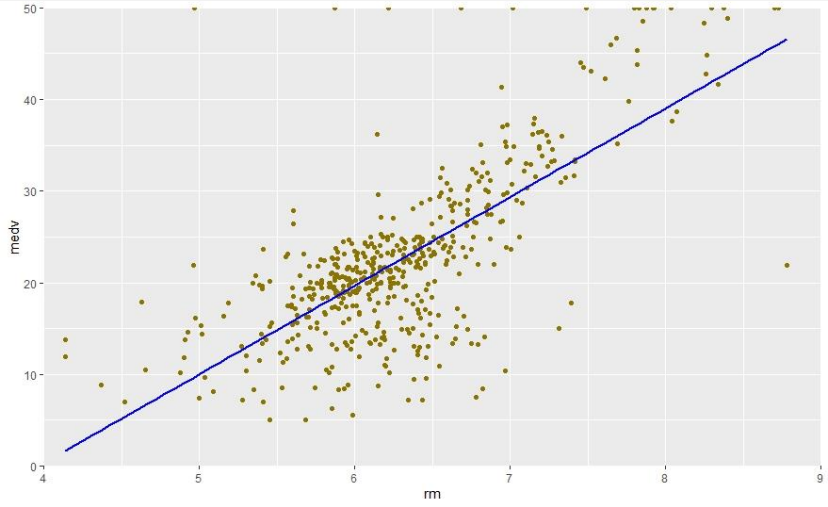
Data ข้อมูลของเรา



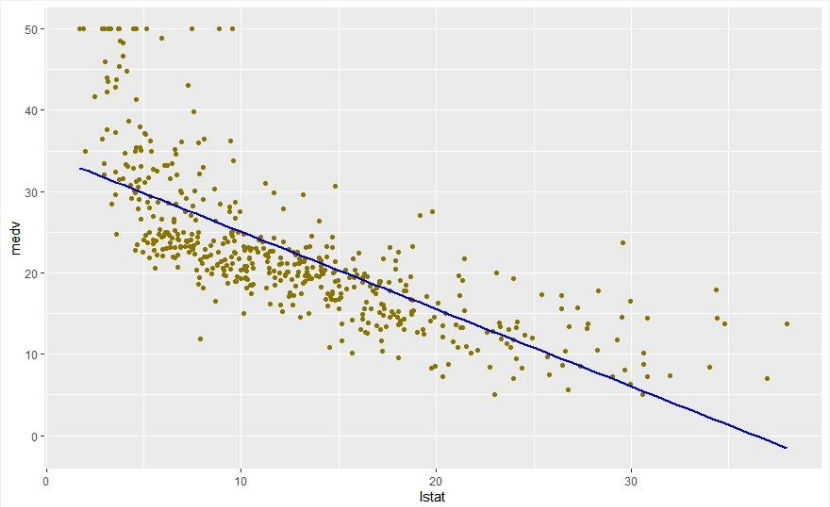
Linear Regression model

Linear Regression Model (Fundamental)

- สร้าง <u>เส้นตรง</u> ที่ fit ข้อมูลได้ดีที่สุด
- นำไป predict ข้อมูลที่ไม่เคยเห็นมาก่อน
- <u>เส้นตรง</u>ของเรา มีสมมุติฐาน (assumption) มาจาก ข้อมูลของเรามีความสัมพันธ์เชิงเส้นตรง



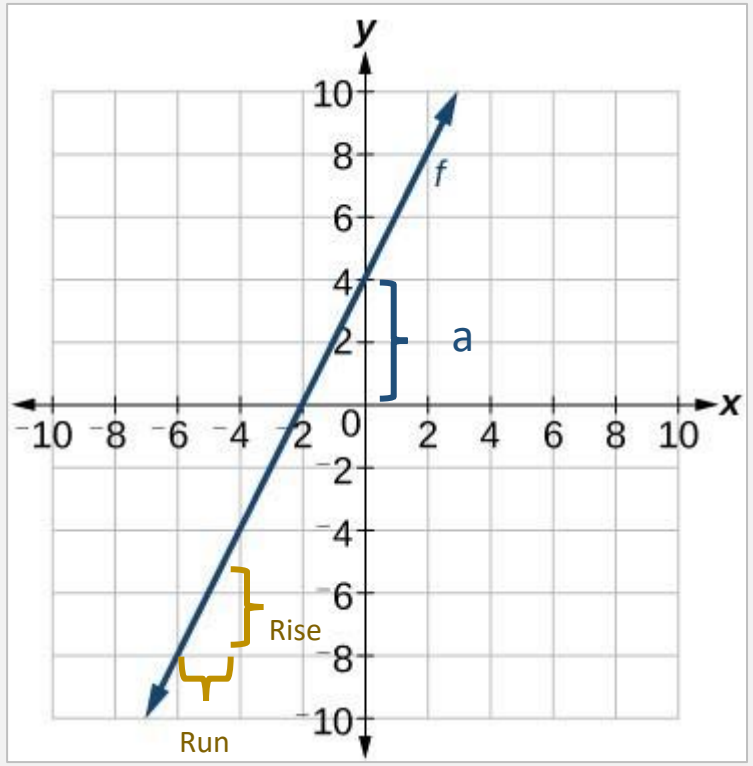
ความสัมพันธ์เชิงบวก (Positive relationship)



ความสัมพันธ์เชิงบวก (negative relationship)

Linear Regression Model (Fundamental)

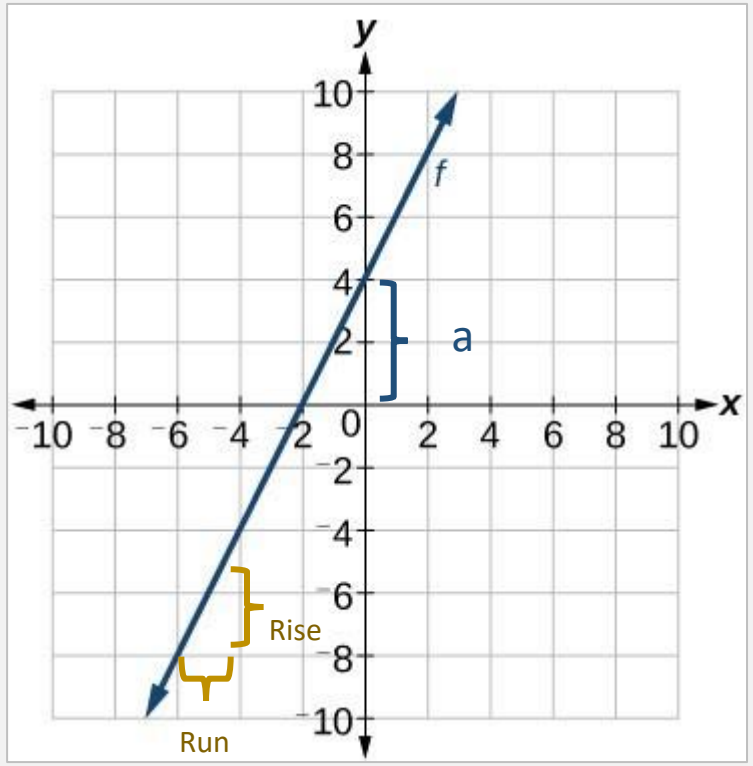
- สมการ เส้นตรงทั่วไป : y = a + bx
- a : intercept
- b : slope (Rise ÷ Run)



Linear Regression Model (Fundamental)

- สมการ เส้นตรงทั่วไป : y = a + bx
- a : intercept
- b : slope (Rise ÷ Run)

$$y = b_0 + b_1 * x_1$$



Linear regression theory (Equation)

Simple Linear Regression

$$y = b_0 + b_1 * x_1$$

Multiple Linear Regression

$$y = b_0 + b_1 * x_1 + b_2 * x_2 + \ldots + b_n * x_n$$

Linear regression theory (Dataset Introduction)

```
library(mlbench)
data("BostonHousing")
str(BostonHousing)

'data.frame':	506 obs. of  14 variables:
 $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
 $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
 $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
 $ chas   : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
 $ rm     : num  6.58 6.42 7.18 7 7.15 ...
 $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
 $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
 $ rad    : num  1 2 2 3 3 3 5 5 5 5 ...
 $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
 $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
 $ b      : num  397 397 393 395 397 ...
 $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
 $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

ราคาบ้านโดยเฉลี่ย (1,000 USD) (medv) | Y | Respond variable | Outcome variable

จำนวนห้องโดยเฉลี่ย (rm) | X | Factor | Features | predictor variable

Linear regression theory (Simple Linear Regression)

Simple Linear Regression

$$y = b_0 + b_1 * x_1$$

ราคาบ้านโดยเฉลี่ย (1,000 USD) (medv)	จำนวนห้องโดยเฉลี่ย (rm)
24	6.575
21.6	6.421
34.7	7.185
33.4	6.998
36.2	7.147
28.7	6.43
22.9	6.012
27.1	6.172
16.5	5.631
18.9	6.004
15	6.377
18.9	6.009
21.7	5.889
20.4	5.949
18.2	6.096
.....	.....
.....	.....

medv = b0 + b1*Rm

ราคาบ้านโดยเฉลี่ย (1,000 USD) (medv) | Y | Respond variable | Outcome variable

จำนวนห้องโดยเฉลี่ย (rm) | X | Factor | Features | predictor variable

Linear regression theory (Simple Linear Regression)

Simple Linear Regression

$$y = b_0 + b_1 * x_1$$

ราคาบ้านโดยเฉลี่ย (1,000 USD) (medv)	จำนวนห้องโดยเฉลี่ย (rm)
24	6.575
21.6	6.421
34.7	7.185
33.4	6.998
36.2	7.147
28.7	6.43
22.9	6.012
27.1	6.172
16.5	5.631
18.9	6.004
15	6.377
18.9	6.009
21.7	5.889
20.4	5.949
18.2	6.096
.....	.....
.....	.....

medv = b0 + b1*Rm

ราคาบ้านโดยเฉลี่ย (1,000 USD) (medv) | Y | Respond variable | Outcome variable

จำนวนห้องโดยเฉลี่ย (rm) | X | Factor | Features | predictor variable

Linear regression theory (Multiple Linear Regression)

Multiple Linear Regression

$$y = b_0 + b_1 * x_1 + b_2 * x_2 + \ldots + b_n * x_n$$

medv	rm	age	dis	ptratio	b	lstat
24	6.575	65.2	4.09	15.3	396.9	4.98
21.6	6.421	78.9	4.9671	17.8	396.9	9.14
34.7	7.185	61.1	4.9671	17.8	392.83	4.03
33.4	6.998	45.8	6.0622	18.7	394.63	2.94
36.2	7.147	54.2	6.0622	18.7	396.9	5.33
28.7	6.43	58.7	6.0622	18.7	394.12	5.21
22.9	6.012	66.6	5.5605	15.2	395.6	12.43
27.1	6.172	96.1	5.9505	15.2	396.9	19.15
16.5	5.631	100	6.0821	15.2	386.63	29.93
18.9	6.004	85.9	6.5921	15.2	386.71	17.1
15	6.377	94.3	6.3467	15.2	392.52	20.45
18.9	6.009	82.9	6.2267	15.2	396.9	13.27
21.7	5.889	39	5.4509	15.2	390.5	15.71
20.4	5.949	61.8	4.7075	21	396.9	8.26
18.2	6.096	84.5	4.4619	21	380.02	10.26
...	...	...	...	...	...	...
...	...	...	...	...	...	...

medv = b0 + b1*Rm +b2*age + b3*dis + b4*ptratio
+b5*b + b6*lstat

ราคาบ้านโดยเฉลี่ย (1,000 USD) | Y | Respond variable | Outcome variable

(rm , age, dis , ptratio , b , lstat) | X | Factor | Features | predictor variable

Linear regression theory (Linear Regression R code)

```
linear_model <- lm(medv ~ rm , data = BostonHousing)
summary(linear_model)
```

```
Call:
lm(formula = medv ~ rm, data = BostonHousing)

Residuals:
    Min      1Q  Median      3Q     Max 
-23.346  -2.547   0.090   2.986  39.433 

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)    
(Intercept)          -34.671      2.650  -13.08   <2e-16 ***
rm                     9.102      0.419   21.72   <2e-16 ***
---
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.616 on 504 degrees of freedom
Multiple R-squared:  0.4835,	Adjusted R-squared:  0.4825 
F-statistic: 471.8 on 1 and 504 DF,  p-value: < 2.2e-16
```

Column (Y) ~ Column (X)

Y = b0 + b1*X1

Medv = -34.671 + 9.102*rm

Linear regression theory (Linear Regression R code caret)

Caret version

```
library(caret)
linear_model_caret <- train(
                            form = medv ~ rm ,
                            data = BostonHousing,
                            method = 'lm’
                            )
summary(linear_model_caret)

Call:
lm(formula = .outcome ~ ., data = dat)

Residuals:
    Min      1Q  Median      3Q     Max 
-23.346  -2.547   0.090   2.986  39.433 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  -34.671      2.650  -13.08   <2e-16 ***
rm             9.102      0.419   21.72   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.616 on 504 degrees of freedom
Multiple R-squared:  0.4835,	Adjusted R-squared:  0.4825 
F-statistic: 471.8 on 1 and 504 DF,  p-value: < 2.2e-16
```

train(
form = สูตร Y กับ X
method = model ที่ใช้,
data = ข้อมูลที่ใช้ Train model ,
trControl = ใช้ในการ control training เช่น cross validation,
tuneGrid = ใช้ในการ tuning
)

SIMPLE LINEAR REGRESION (START STRONG !!!)

Linear regression theory (Simple Linear Regression)

Simple Linear Regression

$$y = b_0 + b_1 * x_1$$

ราคาบ้านโดยเฉลี่ย (1,000 USD) (medv)	จำนวนห้องโดยเฉลี่ย (rm)
24	6.575
21.6	6.421
34.7	7.185
33.4	6.998
36.2	7.147
28.7	6.43
22.9	6.012
27.1	6.172
16.5	5.631
18.9	6.004
15	6.377
18.9	6.009
21.7	5.889
20.4	5.949
18.2	6.096
.....	.....
.....	.....

Medv = b0 + b1*Rm

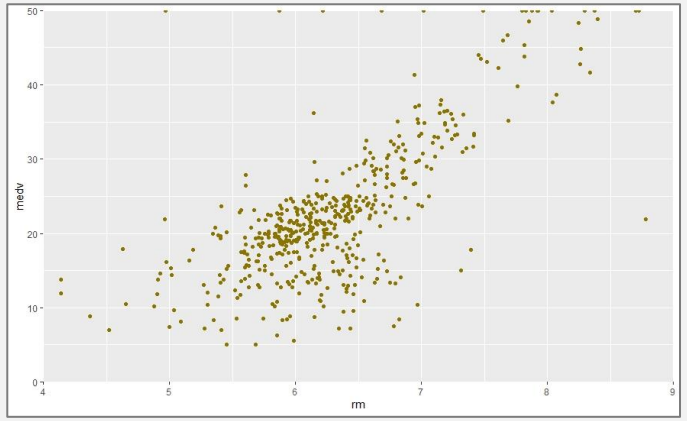
ราคาบ้านโดยเฉลี่ย (1,000 USD) (medv) | Y | Respond variable | Outcome variable

จำนวนห้องโดยเฉลี่ย (rm) | X | Factor | Features | predictor variable

Linear regression theory (Simple Linear Regression)

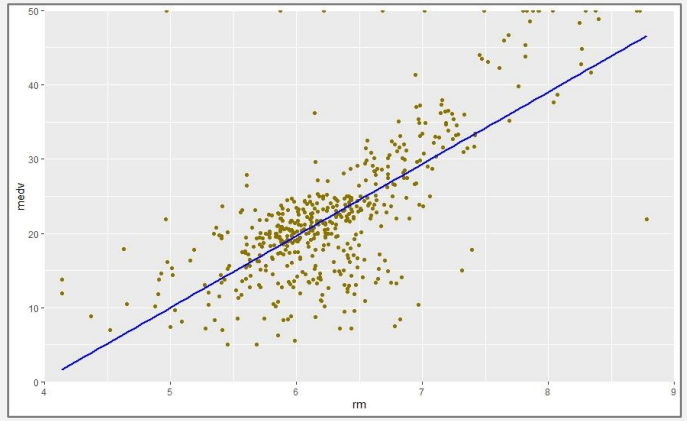
ราคาบ้านโดยเฉลี่ย (1,000 USD) (medv)	จำนวนห้องโดยเฉลี่ย (rm)
24	6.575
21.6	6.421
34.7	7.185
33.4	6.998
36.2	7.147
28.7	6.43
22.9	6.012
27.1	6.172
16.5	5.631
18.9	6.004
15	6.377
18.9	6.009
21.7	5.889
20.4	5.949
18.2	6.096
.....	.....
.....	.....

Data ข้อมูล



หาเส้นตรงยังไง??

เส้นตรงที่ fit ข้อมูลได้ดีที่สุด

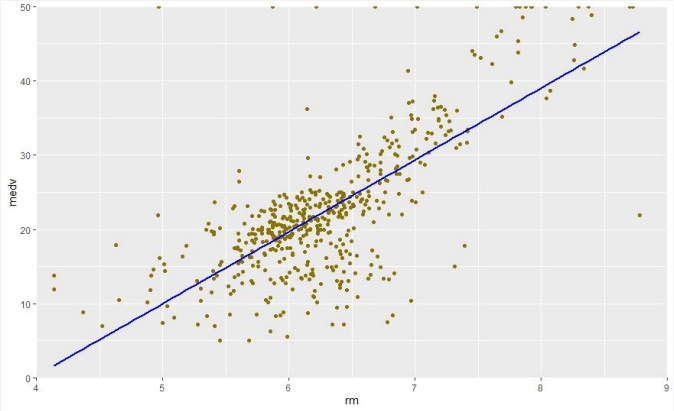


Linear regression theory (Simple Linear Regression)

Estimate ตัว b0 b1 จาก data เราแล้วนำไปลาก เส้นตรง

$$y = b_0 + b_1 * x_1$$

เส้นตรงที่ fit ข้อมูลได้ดีที่สุด



ตัว b0 b1 ต้องให้เส้นตรงที่มีค่า sum of squared error ที่น้อยที่สุด !!!!!

TERM DEFINITION

Linear regression theory (Term definition)

สมการจุด
data point

$$Y_i = b_0 + b_1 * X_i + u_i$$

สมการเส้น
linear regression line

$$Y_Pred_i = b_0 + b_1 * X_i$$

Yi คือข้อมูลที่เราต้องการ predict
ยกตัวอย่างเช่น ราคาบ้าน

Xi คือ ข้อมูลที่เป็น factor / feature ในการช่วย predict Y
ยกตัวอย่างเช่น จำนวนห้องในบ้านนั้น

b0 คือ intercept ที่เราได้มาจากการ ลากเส้นที่ดีที่สุดให้ข้อมูล

b1 คือ slope ที่เราได้มาจากการ ลากเส้นที่ดีที่สุดให้ข้อมูล

Y_Predi คือ ข้อมูลที่ predict ออกมาจากการที่เราใส่ Xi เข้าไป (หรือจุดบอกเส้นตรง)

Ui คือ ระยะห่างระหว่าง เส้นตรง และ จุดdata point

Linear regression theory (Term definition)

สมการจุด
data point

$$Y_i = b0 + b1 * X_i + ui$$

สมการเส้น
linear regression line

$$Y_Predi = b0 + b1 * X_i$$

ราคาบ้านโดยเฉลี่ย (1,000 USD) (medv)	จำนวนห้องโดยเฉลี่ย (rm)
24	6.575
21.6	6.421
34.7	7.185
33.4	6.998
36.2	7.147
28.7	6.43
22.9	6.012
27.1	6.172
16.5	5.631
18.9	6.004
15	6.377
18.9	6.009
21.7	5.889
20.4	5.949
18.2	6.096
.....	.....
.....	.....

Yi คือข้อมูลที่เราต้องการ predict
ยกตัวอย่างเช่น ราคาบ้าน

Xi คือ ข้อมูลที่เป็น factor / feature ในการช่วย predict Y
ยกตัวอย่างเช่น จำนวนห้องในบ้านนั้น

b0 คือ intercept ที่เราได้มาจากการ ลากเส้นที่ดีที่สุดให้ข้อมูล

b1 คือ slope ที่เราได้มาจากการ ลากเส้นที่ดีที่สุดให้ข้อมูล

Y_Predi คือ ข้อมูลที่ predict ออกมาจากการที่เราใส่ Xi เข้าไป (หรือจุดบอกเส้นตรง)

Ui คือ ระยะห่างระหว่าง เส้นตรง และ จุดdata point

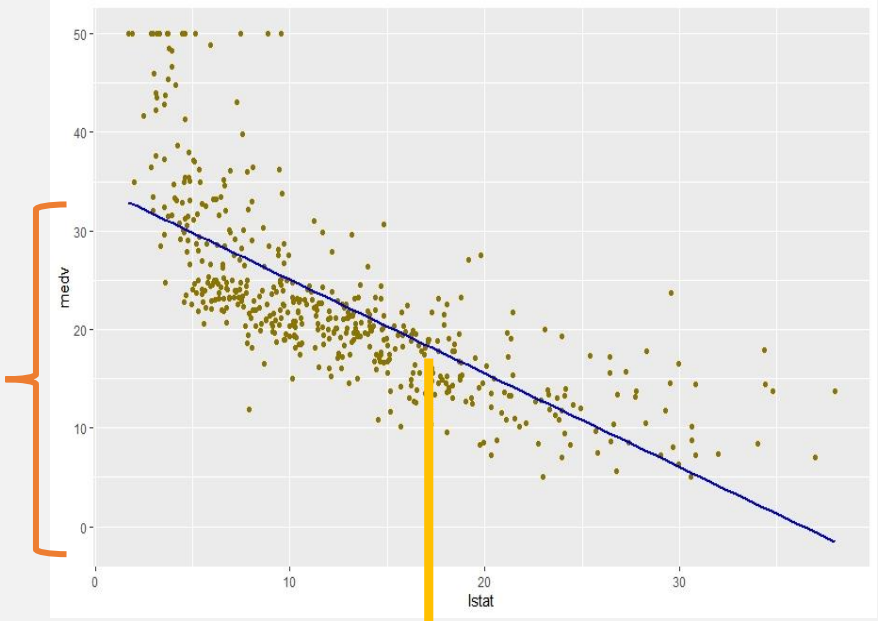
Linear regression theory (Term definition)

สมการจุด
data point

$$Yi = b0 + b1*Xi + ui$$

สมการเส้น
linear regression line

$$Y_Predi = b0 + b1*Xi$$



B0
34.55

b1
-0.95

Yi คือข้อมูลที่เราต้องการ predict
ยกตัวอย่างเช่น ราคาบ้าน

Xi คือ ข้อมูลที่เป็น factor / feature ในการช่วย predict Y
ยกตัวอย่างเช่น จำนวนห้องในบ้านนั้น

b0 คือ intercept ที่เราได้มาจากการ ลากเส้นที่ดีที่สุดให้ข้อมูล

b1 คือ slope ที่เราได้มาจากการ ลากเส้นที่ดีที่สุดให้ข้อมูล

Y_Predi คือ ข้อมูลที่ predict ออกมาจากการที่เราใส่ Xi เข้าไป (หรือจุดบอกเส้นตรง)

Ui คือ ระยะห่างระหว่าง เส้นตรง และ จุดdata point

Linear regression theory (Term definition)

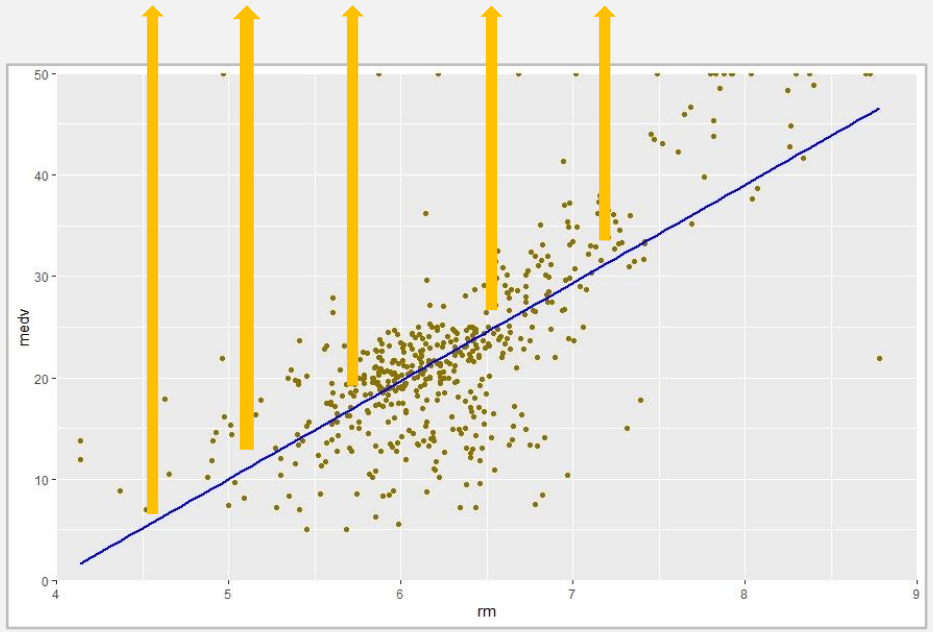
สมการจุด
data point

$$Y_i = b0 + b1 * X_i + u_i$$

สมการเส้น
linear regression line

$$\mathbf{Y_Pred_i} = b0 + b1 * X_i$$

ทั้งเส้นคือ Y_Predi



Yi คือข้อมูลที่เราต้องการ predict
ยกตัวอย่างเช่น ราคาบ้าน

Xi คือ ข้อมูลที่เป็น factor / feature ในการช่วย predict Y
ยกตัวอย่างเช่น จำนวนห้องในบ้านนั้น

b0 คือ intercept ที่เราได้มาจากการ ลากเส้นที่ดีที่สุดให้ข้อมูล

b1 คือ slope ที่เราได้มาจากการ ลากเส้นที่ดีที่สุดให้ข้อมูล

Y_Predi คือ ข้อมูลที่ predict ออกมาจากการที่เราใส่ Xi เข้าไป (หรือจุดบอกเส้นตรง)

Ui คือ ระยะห่างระหว่าง เส้นตรง และ จุดdata point

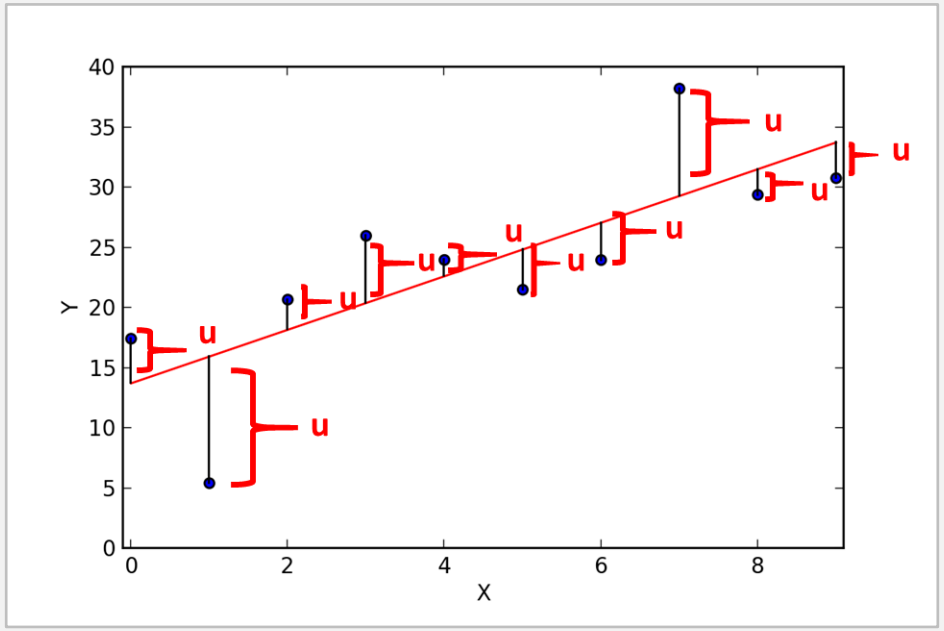
Linear regression theory (Term definition)

สมการจุด
data point

$$Y_i = b_0 + b_1 * X_i + u_i$$

สมการเส้น
linear regression line

$$Y_Predi = b_0 + b_1 * X_i$$



Yi คือข้อมูลที่เราต้องการ predict
ยกตัวอย่างเช่น ราคาบ้าน

Xi คือ ข้อมูลที่เป็น factor / feature ในการช่วย predict Y
ยกตัวอย่างเช่น จำนวนห้องในบ้านนั้น

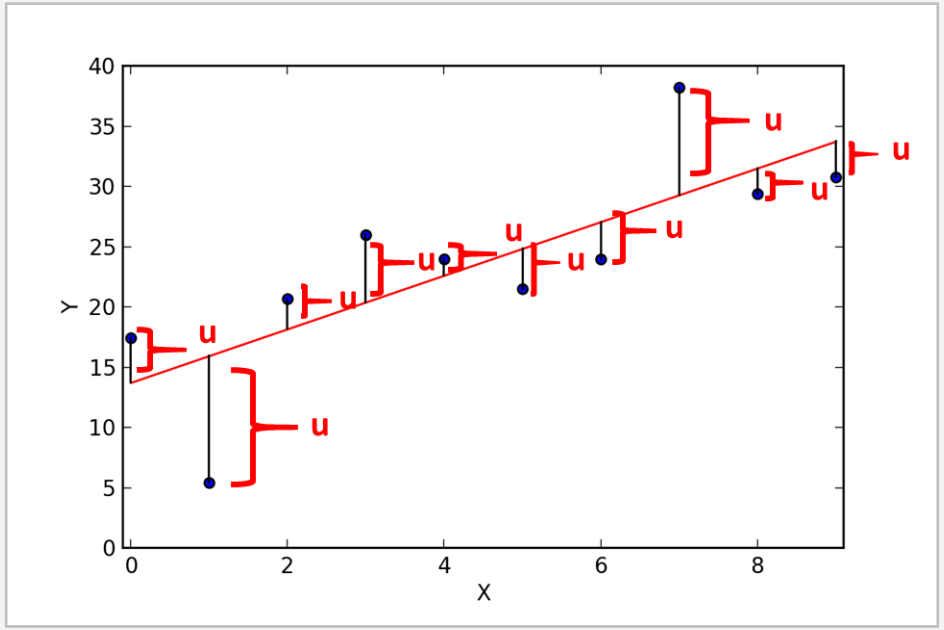
b0 คือ intercept ที่เราได้มาจากการ ลากเส้นที่ดีที่สุดให้ข้อมูล

b1 คือ slope ที่เราได้มาจากการ ลากเส้นที่ดีที่สุดให้ข้อมูล

Y_Predi คือ ข้อมูลที่ predict ออกมาจากการที่เราใส่ Xi เข้าไป (หรือจุดบอกเส้นตรง)

Ui คือ ระยะห่างระหว่าง เส้นตรง และ จุดdata point

Linear regression theory (Regression Cost)



error หรือ residual คือ ระยะห่าง ระหว่างรายจุดข้อมูล กับ เส้นสมการที่เราสร้างขึ้นมา

Linear Regression Cost $= \sum (\hat{y}_i - y_i)$ ^2

เรียกว่า ***SUM of Squared error***

ตัว b0 b1 ต้องให้เส้นตรงที่มีค่า sum of squared error ที่น้อยที่สุด !!!!!

Linear regression theory (Regression Cost)

- 1. Sum of Squared Error :

$$SS_{(residuals)} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

ยิ่งน้อย ยิ่งดี (Minimize)

- 2. Mean Squared Error:

$$\mathrm{MSE} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

ยิ่งน้อย ยิ่งดี (Minimize)

- 3. R-Squared:

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$$

ยิ่งเยอะ ยิ่งดี (Maximize)

b0 b1 Prove!!! (Optional)

Linear regression theory (Regression beta formula)

$$\boxed{\beta_0 = \bar{Y} - \beta_1 \bar{X}}$$

Where

Y_bar : ค่าเฉลี่ยของ Y

X_bar : ค่าเฉลี่ยของ X

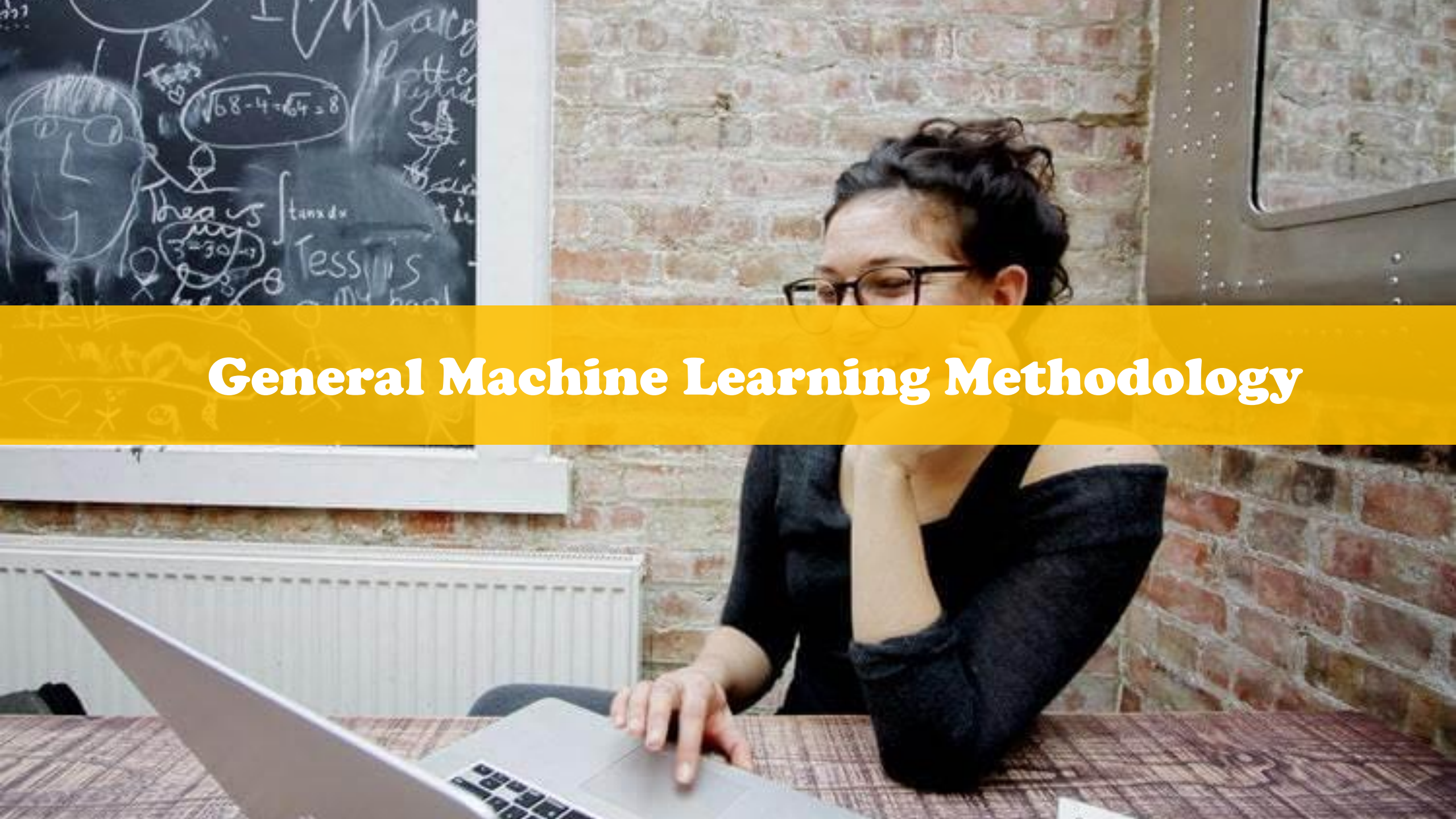
Sigma : ผลรวมต้องแต่ 1 จนถึง จำนวน data point

Xi : ค่า x แต่ละตัว

Yi : ค่า y แต่ละตัว

$$\boxed{\beta_1 = \frac{n\sum y_i x_i - \sum y_i \sum x_i}{n\sum x_i^2 - \left[\sum x_i\right]^2}}$$

$$\boxed{\beta_1 = \frac{\sum (y_i - \bar{y})(x_i - \bar{x})}{\sum (x_i - \bar{x})^2}}$$



General Machine Learning Methodology

Data Split

Validation Set

Hyper Parameter Tuning

Prediction & Evaluation

Optimization

Data Split

Validation Set

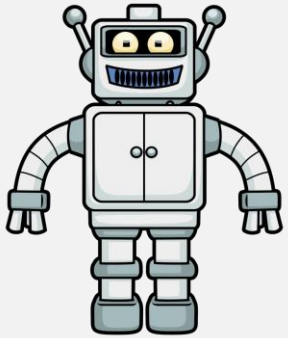
Hyper Parameter Tuning

Prediction & Evaluation

Optimization

General Machine Learning Methodology (Train Test split)

Machine Learning Model
(Ex Linear regression)

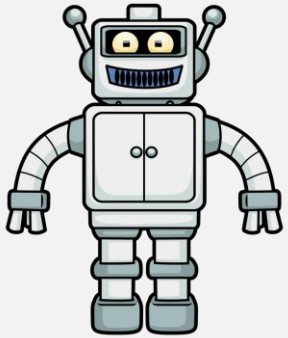


Predict

Unseen Data

General Machine Learning Methodology (Train Test split)

Machine Learning Model
(Ex Linear regression)



Predict

Unseen Data

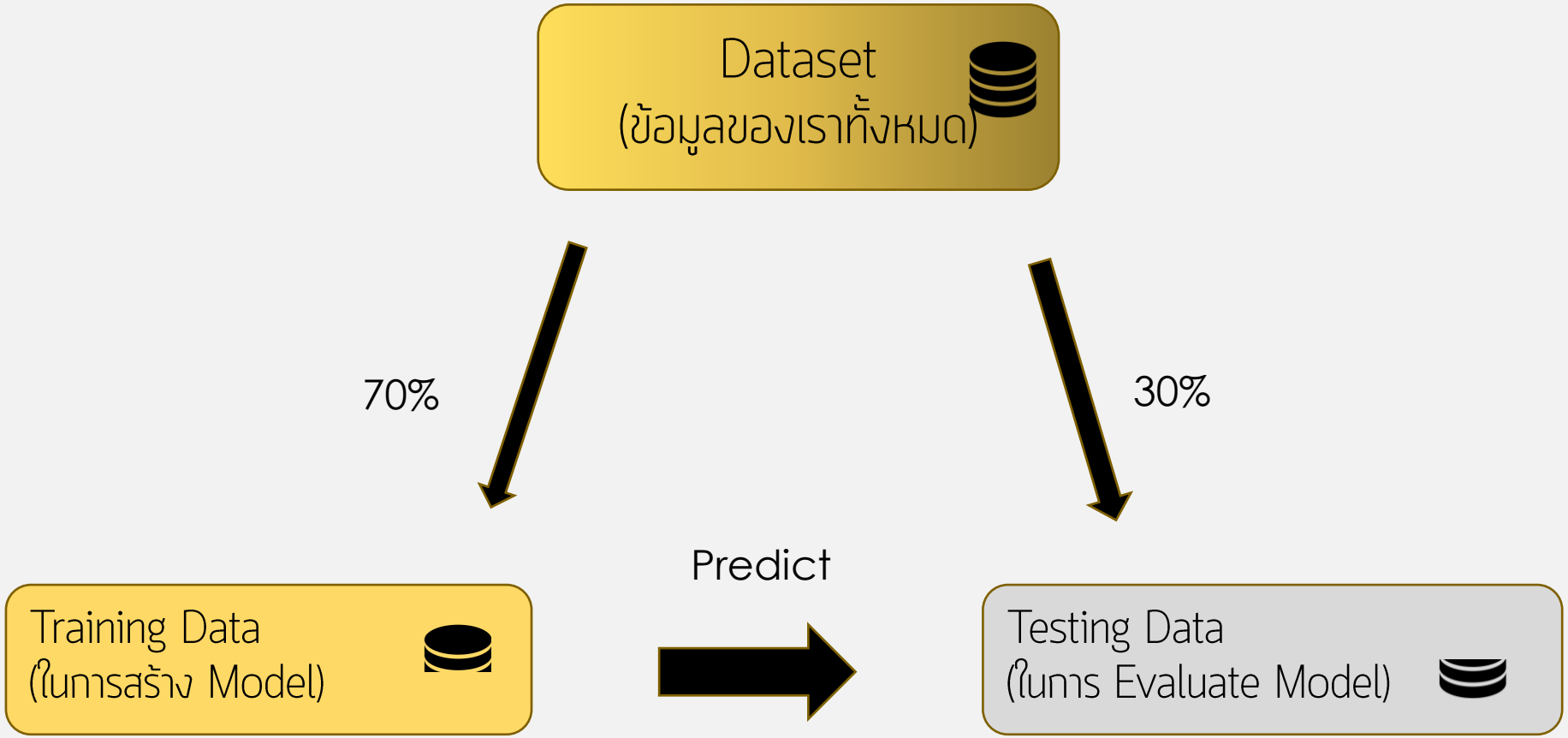
Predict

Training Data
(ในการสร้าง Model)

Testing Data
(ในการ Evaluate Model)

General Machine Learning Methodology (Train Test split)

Assume สัดส่วนการแบ่ง = 70/30



General Machine Learning Methodology (Train Test split)

Assume สัดส่วนการแบ่ง = 70/30

medv	rm
24	6.575
21.6	6.421
34.7	7.185
33.4	6.998
36.2	7.147
28.7	6.43
22.9	6.012
27.1	6.172
16.5	5.631
18.9	6.004

medv	rm
24	6.575
21.6	6.421
34.7	7.185
33.4	6.998
36.2	7.147
28.7	6.43
22.9	6.012
27.1	6.172
16.5	5.631
18.9	6.004

70%

Training Data 70%

medv	rm
24	6.575
34.7	7.185
33.4	6.998
36.2	7.147
28.7	6.43
27.1	6.172
16.5	5.631

30%

Testing Data 30%

medv	rm
21.6	6.421
22.9	6.012
18.9	6.004

General Machine Learning Methodology (Train Test split)

สมมุติ สัดส่วนการแบ่ง = 70/30

medv	rm
24	6.575
21.6	6.421
34.7	7.185
33.4	6.998
36.2	7.147
28.7	6.43
22.9	6.012
27.1	6.172
16.5	5.631
18.9	6.004

medv	rm
24	6.575
21.6	6.421
34.7	7.185
33.4	6.998
36.2	7.147
28.7	6.43
22.9	6.012
27.1	6.172
16.5	5.631
18.9	6.004

70%

Training Data 70%

medv	rm
24	6.575
34.7	7.185
33.4	6.998
36.2	7.147
28.7	6.43
27.1	6.172
16.5	5.631

30%

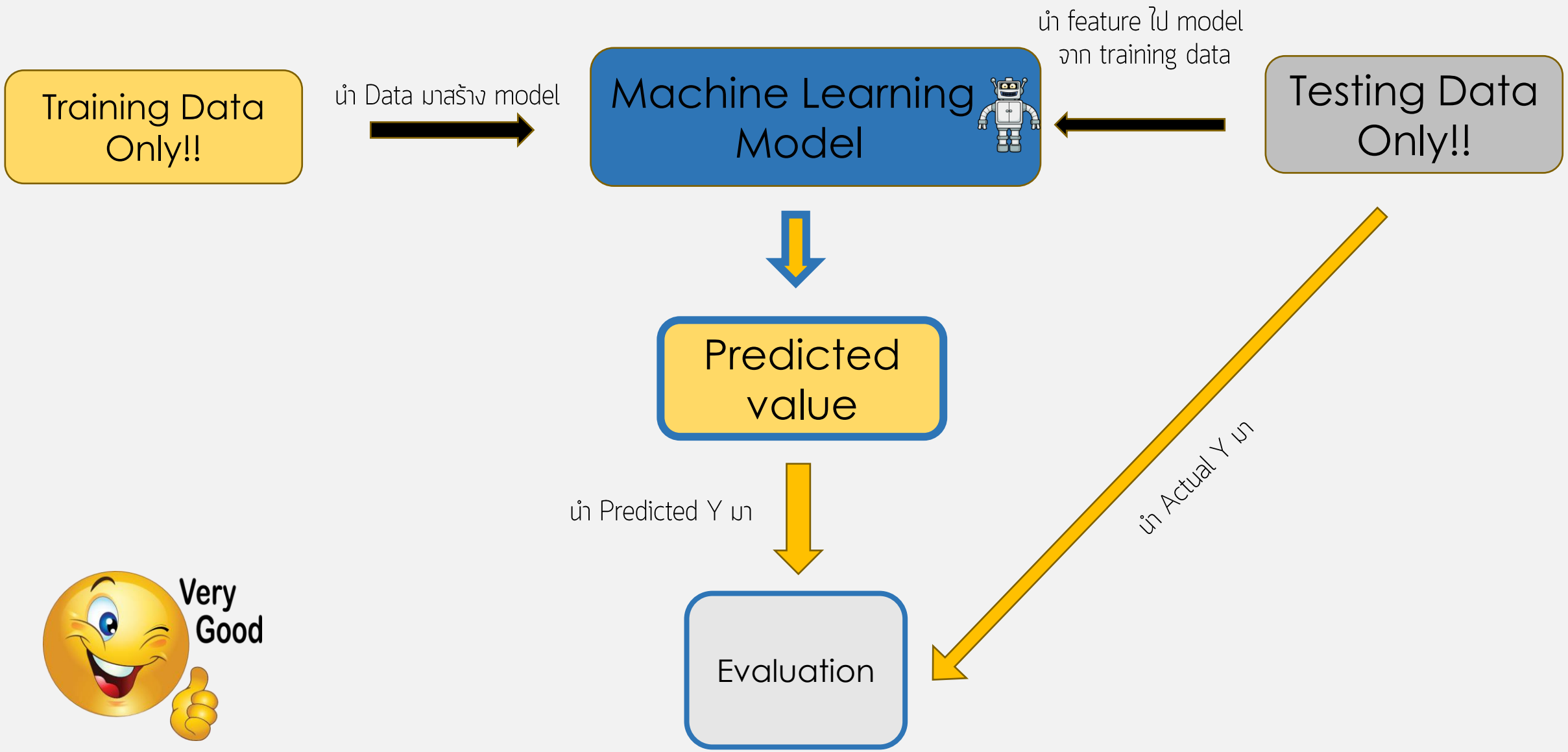
Testing Data 30%

medv	rm
21.6	6.421
22.9	6.012
18.9	6.004

```
library(caret)

num_train <- createDataPartition(y= BostonHousing$medv , p =0.70 , list = FALSE)
train_data <- BostonHousing[num_train,]
test_data <- BostonHousing[-num_train,]
```

y = Column ของ Y
p = สัดส่วนการแบ่งไปใน training data
list = False คือการบังคับไม่ให้ return ค่าเป็น list

General Machine Learning Methodology (Train Test split)



General Machine Learning Methodology (Train Test split Example)

รูปร่าง data Train Test

```
train_data
  medv    rm
1 24.0 6.575
2 34.7 7.185
3 33.4 6.998
4 36.2 7.147
5 28.7 6.430
6 27.1 6.172
7 16.5 5.631
test_data
  medv    rm
1 21.6 6.421
2 22.9 6.012
3 18.9 6.004
```

ทำ linear regression model จาก train_data ได้ machine learning model ชื่อว่า lm_model

```
library(caret)

lm_model <- lm(medv ~ rm, data = train_data)

summary(lm_model)


Call:

lm(formula = medv ~ rm, data = train_data)


Residuals:

      1       2       3       4       5       6       7

-4.4742 -0.6856  0.1331  1.2449  1.8686  3.1918 -1.2786


Coefficients:

            Estimate Std. Error t value Pr(>|t|)

(Intercept)  -46.021     12.942  -3.556  0.01628 *

rm            11.330      1.957   5.789  0.00217 **

---

Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1


Residual standard error: 2.734 on 5 degrees of freedom

Multiple R-squared:  0.8702,	Adjusted R-squared:  0.8442

F-statistic: 33.51 on 1 and 5 DF,  p-value: 0.002166
```

นำ model ไป predict กับ test_data แล้วจะได้ค่า prediction มาวัดกับ actual value ว่าใกล้เคียงกันไหม

```
test_data['pred'] <- predict(lm_model ,
newdata = test_data)

test_data
  medv    rm     pred
1 21.6 6.421 26.72941
2 22.9 6.012 22.09540
3 18.9 6.004 22.00476
RMSE(test_data$medv , test_data$pred)
[1] 3.492742
```

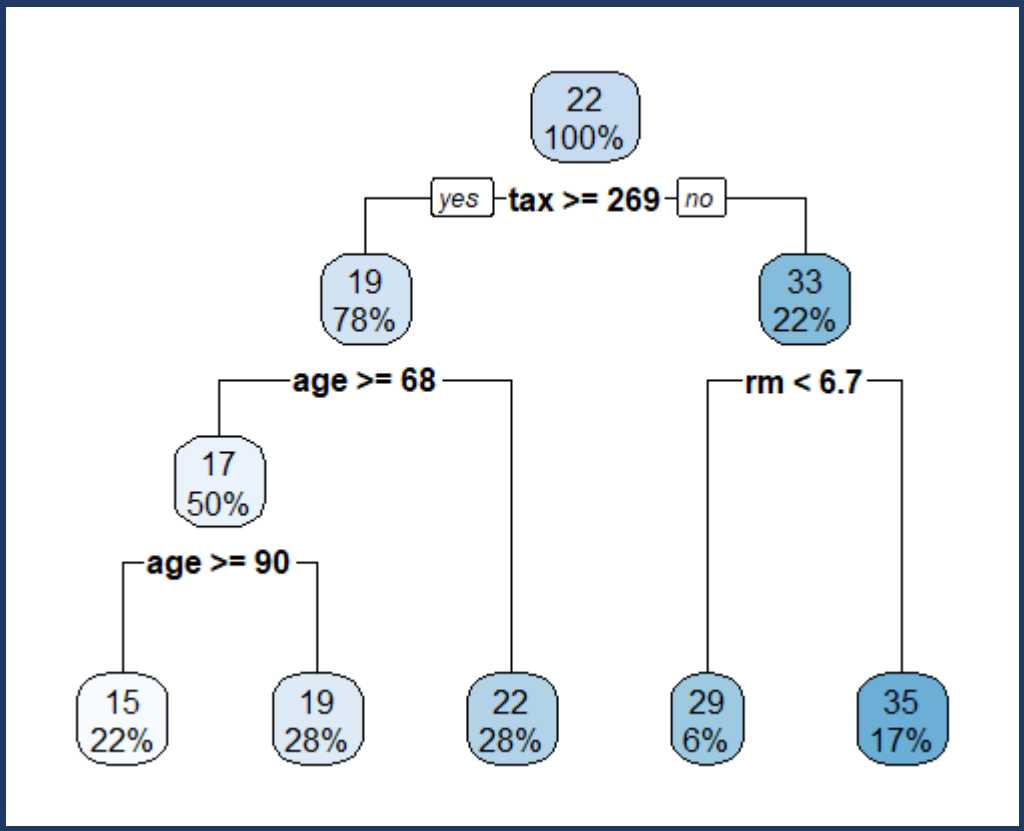
Data Split

Validation Set

Hyper Parameter Tuning

Prediction & Evaluation

Optimization

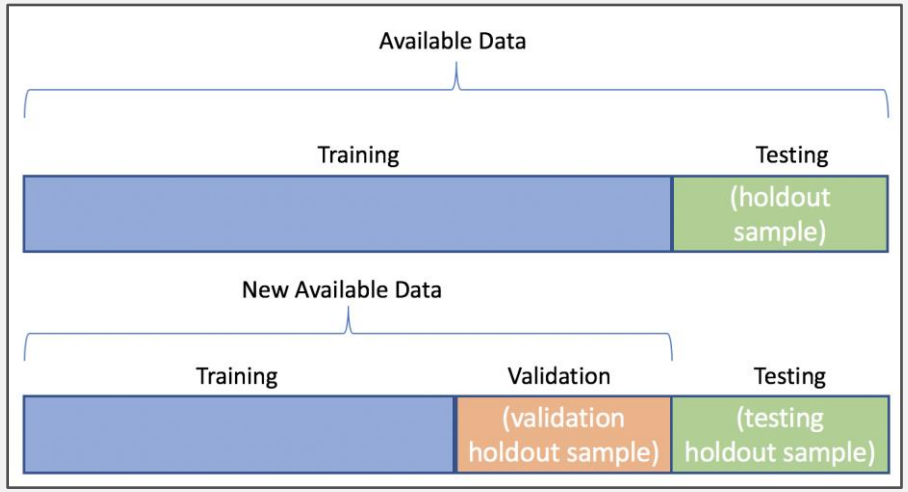


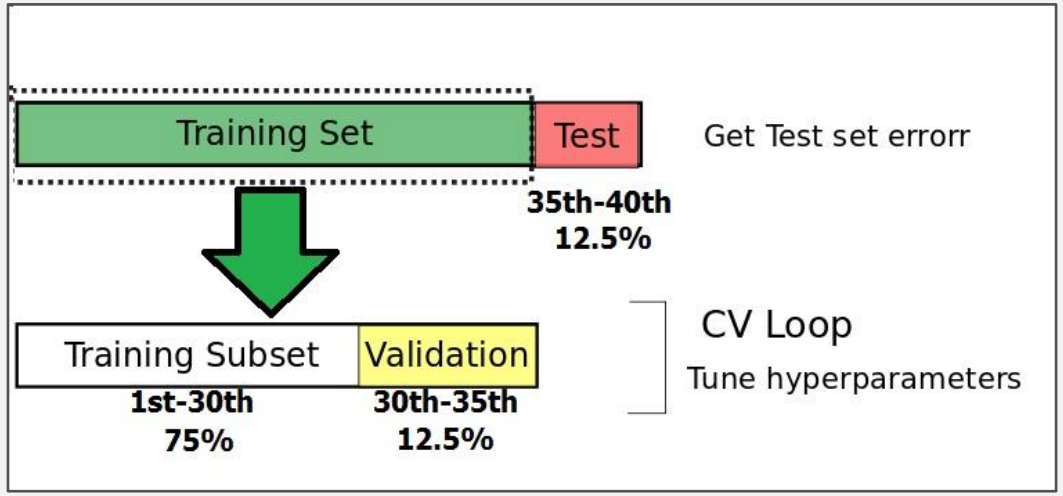
Brief introduction to tree regression

General Machine Learning Methodology (Validation Set)

- นอกจาก Training set และ Testing set แล้ว เรายังสามารถแบ่ง Dataset เพิ่มจาก Training set ได้อีก ซึ่งคือ Validation Set
- Validation Set จะทำงานคล้ายๆ Testing set ซึ่งคือไว้วัดผลของ Model แต่การวัดผลนี้จะเป็นการวัดผลของ hyper parameter ที่เรา set ไว้
- ตัวอย่าง Tree Regression : hyperparameter คือ **cp**
  - 1. **set cp = 0.001** แล้ว Train model จาก **Training set** วัดผล RMSE (**RMSE1**) จาก **Validation Set**
  - 2. **set cp = 0.005** แล้ว Train model จาก **Training set** วัดผล RMSE (**RMSE2**) จาก **Validation Set**
  - 3. เทียบ **RMSE1 vs RMSE2** สมมุติ RMSE1 น้อยกว่า เราก็จะสรุปว่าใช้ cp = 0.001 ให้ผลดีสุดใน validation Set
  - 4. นำ **model (cp = 0.001) ไป predict Testing Set** เพื่อวัดผลครั้งสุดท้าย

  เราสามารถสร้าง validation set โดยการ cross validation ได้
  เป้าหมายการทำแบบนี้เพื่อ ให้ Model Generalized ได้มากที่สุด **(ลดการ overfit)**





General Machine Learning Methodology (Validation Set)

All data

medv	age	rm	lstat	tax
19.9	91.1	6.251	14.19	666
24	61.5	6.086	10.88	307
38.7	76	8.069	4.21	276
22.6	17.2	6.333	7.34	430
28.1	32.3	6.375	9.38	277
15.7	95.6	5.613	27.26	188
20	84.4	6.081	14.7	666
19.3	88.6	5.404	23.98	277
23.6	84.1	6.416	9.04	296
20.3	76.7	5.972	9.97	304
11.8	94.8	6.459	23.98	666
22	40.4	6.59	9.5	329
24.4	58.7	6.538	7.73	223
31.7	76.9	7.412	5.25	307
27.5	87.9	3.561	7.12	666
22.2	52.9	6.358	11.22	300
26.6	34.5	6.939	5.89	335
20.7	38.5	6.31	6.75	224
35.1	32.2	6.812	4.85	216
18.8	92.9	5.913	16.21	432
19.9	84	6.167	16.29	666
19.5	91.9	5.836	18.66	384
15.1	97	5.713	17.11	666
8.8	100	5.957	20.62	666

Training_dataset

medv	age	rm	lstat	tax
38.7	76	8.069	4.21	276
24.4	58.7	6.538	7.73	223
26.6	34.5	6.939	5.89	335
11.8	94.8	6.459	23.98	666
28.1	32.3	6.375	9.38	277
22.6	17.2	6.333	7.34	430
19.9	84	6.167	16.29	666
15.7	95.6	5.613	27.26	188
23.6	84.1	6.416	9.04	296
35.1	32.2	6.812	4.85	216
19.5	91.9	5.836	18.66	384
22.2	52.9	6.358	11.22	300
18.8	92.9	5.913	16.21	432
20.3	76.7	5.972	9.97	304
19.9	91.1	6.251	14.19	666
27.5	87.9	3.561	7.12	666
20	84.4	6.081	14.7	666
22	40.4	6.59	9.5	329

validation_dataset

medv	age	rm	lstat	tax
31.7	76.9	7.412	5.25	307
20.7	38.5	6.31	6.75	224
8.8	100	5.957	20.62	666

Testing_dataset

medv	age	rm	lstat	tax
24	61.5	6.086	10.88	307
19.3	88.6	5.404	23.98	277
15.1	97	5.713	17.11	666

General Machine Learning Methodology (Validation Set)

All data

medv	age	rm	lstat	tax
19.9	91.1	6.251	14.19	666
24	61.5	6.086	10.88	307
38.7	76	8.069	4.21	276
22.6	17.2	6.333	7.34	430
28.1	32.3	6.375	9.38	277
15.7	95.6	5.613	27.26	188
20	84.4	6.081	14.7	666
19.3	88.6	5.404	23.98	277
23.6	84.1	6.416	9.04	296
20.3	76.7	5.972	9.97	304
11.8	94.8	6.459	23.98	666
22	40.4	6.59	9.5	329
24.4	58.7	6.538	7.73	223
31.7	76.9	7.412	5.25	307
27.5	87.9	3.561	7.12	666
22.2	52.9	6.358	11.22	300
26.6	34.5	6.939	5.89	335
20.7	38.5	6.31	6.75	224
35.1	32.2	6.812	4.85	216
18.8	92.9	5.913	16.21	432
19.9	84	6.167	16.29	666
19.5	91.9	5.836	18.66	384
15.1	97	5.713	17.11	666
8.8	100	5.957	20.62	666

Training_dataset

medv	age	rm	lstat	tax
38.7	76	8.069	4.21	276
24.4	58.7	6.538	7.73	223
26.6	34.5	6.939	5.89	335
11.8	94.8	6.459	23.98	666
28.1	32.3	6.375	9.38	277
22.6	17.2	6.333	7.34	430
19.9	84	6.167	16.29	666
15.7	95.6	5.613	27.26	188
23.6	84.1	6.416	9.04	296
35.1	32.2	6.812	4.85	216
19.5	91.9	5.836	18.66	384
22.2	52.9	6.358	11.22	300
18.8	92.9	5.913	16.21	432
20.3	76.7	5.972	9.97	304
19.9	91.1	6.251	14.19	666
27.5	87.9	3.561	7.12	666
20	84.4	6.081	14.7	666
22	40.4	6.59	9.5	329

validation_dataset

medv	age	rm	lstat	tax
31.7	76.9	7.412	5.25	307
20.7	38.5	6.31	6.75	224
8.8	100	5.957	20.62	666

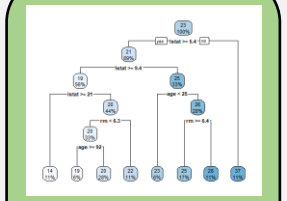
Testing_dataset

medv	age	rm	lstat	tax
24	61.5	6.086	10.88	307
19.3	88.6	5.404	23.98	277
15.1	97	5.713	17.11	666

General Machine Learning Methodology (Validation Set)

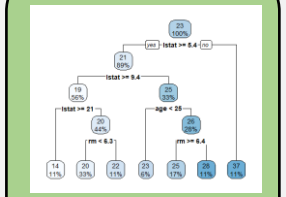
Training_dataset

medv	age	rm	lstat	tax
38.7	76	8.069	4.21	276
24.4	58.7	6.538	7.73	223
26.6	34.5	6.939	5.89	335
11.8	94.8	6.459	23.98	666
28.1	32.3	6.375	9.38	277
22.6	17.2	6.333	7.34	430
19.9	84	6.167	16.29	666
15.7	95.6	5.613	27.26	188
23.6	84.1	6.416	9.04	296
35.1	32.2	6.812	4.85	216
19.5	91.9	5.836	18.66	384
22.2	52.9	6.358	11.22	300
18.8	92.9	5.913	16.21	432
20.3	76.7	5.972	9.97	304
19.9	91.1	6.251	14.19	666
27.5	87.9	3.561	7.12	666
20	84.4	6.081	14.7	666
22	40.4	6.59	9.5	329



MODEL
(CP = 0.001)

medv	age	rm	lstat	tax
38.7	76	8.069	4.21	276
24.4	58.7	6.538	7.73	223
26.6	34.5	6.939	5.89	335
11.8	94.8	6.459	23.98	666
28.1	32.3	6.375	9.38	277
22.6	17.2	6.333	7.34	430
19.9	84	6.167	16.29	666
15.7	95.6	5.613	27.26	188
23.6	84.1	6.416	9.04	296
35.1	32.2	6.812	4.85	216
19.5	91.9	5.836	18.66	384
22.2	52.9	6.358	11.22	300
18.8	92.9	5.913	16.21	432
20.3	76.7	5.972	9.97	304
19.9	91.1	6.251	14.19	666
27.5	87.9	3.561	7.12	666
20	84.4	6.081	14.7	666
22	40.4	6.59	9.5	329



MODEL
(CP = 0.005)

validation_dataset

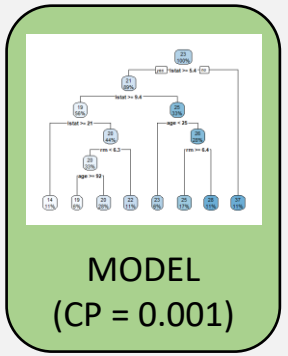
medv	age	rm	lstat	tax
31.7	76.9	7.412	5.25	307
20.7	38.5	6.31	6.75	224
8.8	100	5.957	20.62	666

medv	age	rm	lstat	tax
31.7	76.9	7.412	5.25	307
20.7	38.5	6.31	6.75	224
8.8	100	5.957	20.62	666

General Machine Learning Methodology (Validation Set)

Training_dataset

medv	age	rm	lstat	tax
38.7	76	8.069	4.21	276
24.4	58.7	6.538	7.73	223
26.6	34.5	6.939	5.89	335
11.8	94.8	6.459	23.98	666
28.1	32.3	6.375	9.38	277
22.6	17.2	6.333	7.34	430
19.9	84	6.167	16.29	666
15.7	95.6	5.613	27.26	188
23.6	84.1	6.416	9.04	296
35.1	32.2	6.812	4.85	216
19.5	91.9	5.836	18.66	384
22.2	52.9	6.358	11.22	300
18.8	92.9	5.913	16.21	432
20.3	76.7	5.972	9.97	304
19.9	91.1	6.251	14.19	666
27.5	87.9	3.561	7.12	666
20	84.4	6.081	14.7	666
22	40.4	6.59	9.5	329

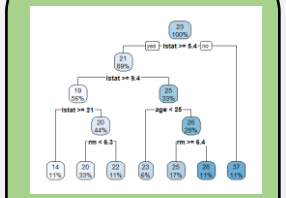


MODEL
(CP = 0.001)

validation_dataset

medv	age	rm	lstat	tax
31.7	76.9	7.412	5.25	307
20.7	38.5	6.31	6.75	224
8.8	100	5.957	20.62	666

medv	age	rm	lstat	tax
38.7	76	8.069	4.21	276
24.4	58.7	6.538	7.73	223
26.6	34.5	6.939	5.89	335
11.8	94.8	6.459	23.98	666
28.1	32.3	6.375	9.38	277
22.6	17.2	6.333	7.34	430
19.9	84	6.167	16.29	666
15.7	95.6	5.613	27.26	188
23.6	84.1	6.416	9.04	296
35.1	32.2	6.812	4.85	216
19.5	91.9	5.836	18.66	384
22.2	52.9	6.358	11.22	300
18.8	92.9	5.913	16.21	432
20.3	76.7	5.972	9.97	304
19.9	91.1	6.251	14.19	666
27.5	87.9	3.561	7.12	666
20	84.4	6.081	14.7	666
22	40.4	6.59	9.5	329



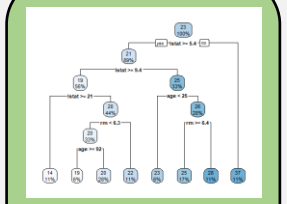
MODEL
(CP = 0.005)

medv	age	rm	lstat	tax
31.7	76.9	7.412	5.25	307
20.7	38.5	6.31	6.75	224
8.8	100	5.957	20.62	666

General Machine Learning Methodology (Validation Set)

Training_dataset

medv	age	rm	lstat	tax
38.7	76	8.069	4.21	276
24.4	58.7	6.538	7.73	223
26.6	34.5	6.939	5.89	335
11.8	94.8	6.459	23.98	666
28.1	32.3	6.375	9.38	277
22.6	17.2	6.333	7.34	430
19.9	84	6.167	16.29	666
15.7	95.6	5.613	27.26	188
23.6	84.1	6.416	9.04	296
35.1	32.2	6.812	4.85	216
19.5	91.9	5.836	18.66	384
22.2	52.9	6.358	11.22	300
18.8	92.9	5.913	16.21	432
20.3	76.7	5.972	9.97	304
19.9	91.1	6.251	14.19	666
27.5	87.9	3.561	7.12	666
20	84.4	6.081	14.7	666
22	40.4	6.59	9.5	329



MODEL
(CP = 0.001)

Final Predict

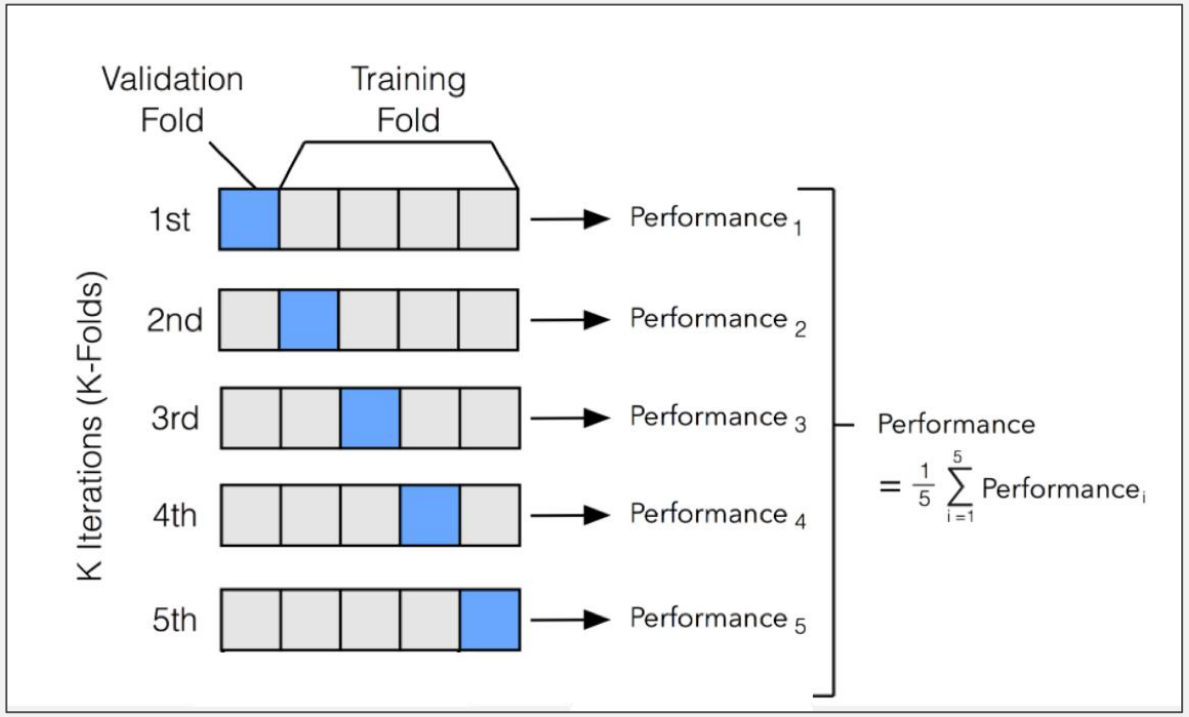
Testing_dataset

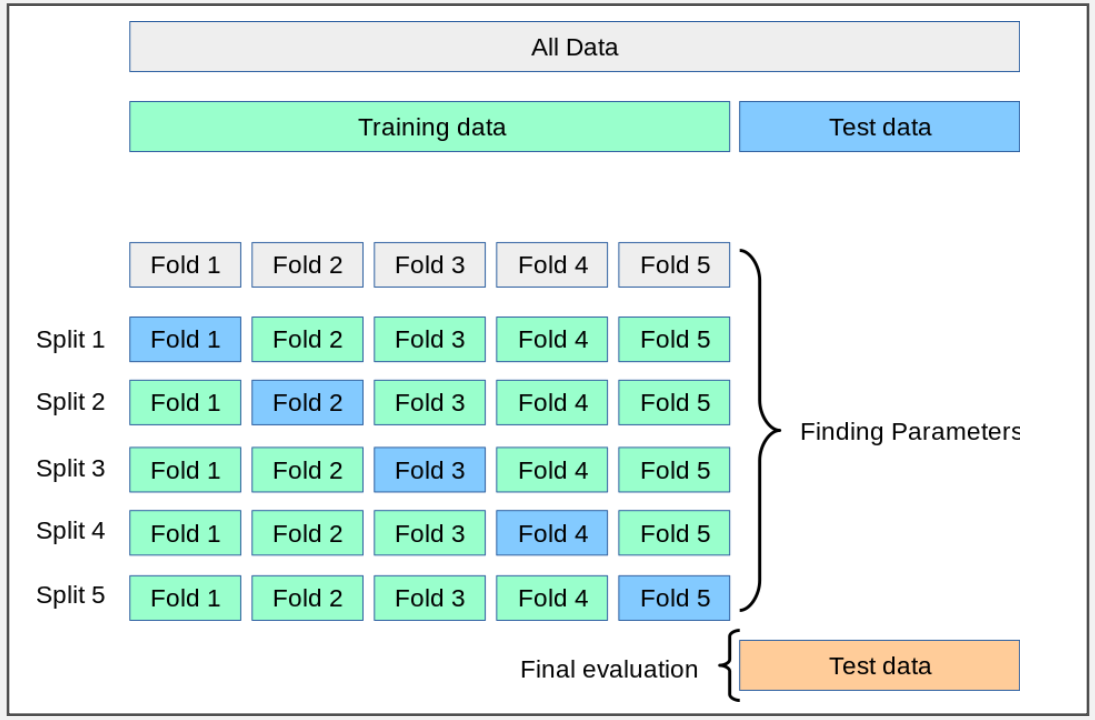
medv	age	rm	lstat	tax
24	61.5	6.086	10.88	307
19.3	88.6	5.404	23.98	277
15.1	97	5.713	17.11	666

Final RMSE = 4.846811

General Machine Learning Methodology (Cross Validation)

- **Cross Validation** ทำโดยการแบ่ง Data set เป็น หลายก้อน (สมมุติว่า 5 ก้อน)
- โดยจะนำ 4 ก้อนไป รวมกันเป็น Training set แล้ว 1 ก้อนที่เหลือไปเป็น Validation set (โดยนำ Training set ไป Train Model และ validation set ไป Evaluate Model)
- หลังจากนั้น สลับก้อน การทำ Training set และ validation set ไปเรื่อยๆ จนครบ 5 ครั้ง แล้วนำ error มา average กัน
- เป้าหมายการทำแบบนี้เพื่อ ให้ Model Generalized ได้มากที่สุด **(ลดการ overfit)**





General Machine Learning Methodology (Cross Validation)

Train and validation Data

medv	age	rm	lstat	tax
24.8	22.9	6.565	9.51	293
15	100	5.757	10.11	666
15.6	98.4	5.757	17.31	437
22.6	17.2	6.333	7.34	430
15.4	100	6.485	18.85	666
48.8	91.5	8.398	5.91	264
11.7	70.6	5.565	17.16	666
21.7	17.5	5.961	9.88	305
14	93.6	6.174	24.16	437
17.2	52.3	5.898	12.67	352
21.4	45	6.286	8.94	398
19.8	97.1	6.474	12.27	384
17.1	28.4	5.706	12.43	304
7	98.3	5.414	23.97	711
9.7	97	5.036	25.68	666
18.6	18.5	5.884	7.79	411
22.2	67.8	6.456	6.73	284
30.1	80.8	6.618	7.6	307
50	89.6	5.875	8.88	666
17.4	36.8	5.594	13.09	345
15.2	91.7	6.142	18.72	307
25	93	6.101	9.81	403
24.5	42.6	5.926	13.59	391
23.3	56	5.888	13.51	276
20	100	5.344	23.09	277

Test Data

medv	age	rm	lstat	tax
13.4	100	5.403	26.82	403
14.5	100	6.072	13.04	307
22.8	82.5	6.635	4.54	304
20.1	83.5	5.983	13.35	711
16.7	93.6	6.484	18.68	666

General Machine Learning Methodology (Cross Validation)

Train and validation Data

สมมุติ
แบ่งเป็น 5
ก้อน

medv	age	rm	lstat	tax
24.8	22.9	6.565	9.51	293
15	100	5.757	10.11	666
15.6	98.4	5.757	17.31	437
22.6	17.2	6.333	7.34	430
15.4	100	6.485	18.85	666
48.8	91.5	8.398	5.91	264
11.7	70.6	5.565	17.16	666
21.7	17.5	5.961	9.88	305
14	93.6	6.174	24.16	437
17.2	52.3	5.898	12.67	352
21.4	45	6.286	8.94	398
19.8	97.1	6.474	12.27	384
17.1	28.4	5.706	12.43	304
7	98.3	5.414	23.97	711
9.7	97	5.036	25.68	666
18.6	18.5	5.884	7.79	411
22.2	67.8	6.456	6.73	284
30.1	80.8	6.618	7.6	307
50	89.6	5.875	8.88	666
17.4	36.8	5.594	13.09	345
15.2	91.7	6.142	18.72	307
25	93	6.101	9.81	403
24.5	42.6	5.926	13.59	391
23.3	56	5.888	13.51	276
20	100	5.344	23.09	277

1

Validation dataset

Training dataset

medv	age	rm	lstat	tax
24.8	22.9	6.565	9.51	293
15	100	5.757	10.11	666
15.6	98.4	5.757	17.31	437
22.6	17.2	6.333	7.34	430
15.4	100	6.485	18.85	666
48.8	91.5	8.398	5.91	264
11.7	70.6	5.565	17.16	666
21.7	17.5	5.961	9.88	305
14	93.6	6.174	24.16	437
17.2	52.3	5.898	12.67	352
21.4	45	6.286	8.94	398
19.8	97.1	6.474	12.27	384
17.1	28.4	5.706	12.43	304
7	98.3	5.414	23.97	711
9.7	97	5.036	25.68	666
18.6	18.5	5.884	7.79	411
22.2	67.8	6.456	6.73	284
30.1	80.8	6.618	7.6	307
50	89.6	5.875	8.88	666
17.4	36.8	5.594	13.09	345
15.2	91.7	6.142	18.72	307
25	93	6.101	9.81	403
24.5	42.6	5.926	13.59	391
23.3	56	5.888	13.51	276
20	100	5.344	23.09	277

2

Training dataset

Validation dataset

Training dataset

medv	age	rm	lstat	tax
24.8	22.9	6.565	9.51	293
15	100	5.757	10.11	666
15.6	98.4	5.757	17.31	437
22.6	17.2	6.333	7.34	430
15.4	100	6.485	18.85	666
48.8	91.5	8.398	5.91	264
11.7	70.6	5.565	17.16	666
21.7	17.5	5.961	9.88	305
14	93.6	6.174	24.16	437
17.2	52.3	5.898	12.67	352
21.4	45	6.286	8.94	398
19.8	97.1	6.474	12.27	384
17.1	28.4	5.706	12.43	304
7	98.3	5.414	23.97	711
9.7	97	5.036	25.68	666
18.6	18.5	5.884	7.79	411
22.2	67.8	6.456	6.73	284
30.1	80.8	6.618	7.6	307
50	89.6	5.875	8.88	666
17.4	36.8	5.594	13.09	345
15.2	91.7	6.142	18.72	307
25	93	6.101	9.81	403
24.5	42.6	5.926	13.59	391
23.3	56	5.888	13.51	276
20	100	5.344	23.09	277

3

Training dataset

Validation dataset

Training dataset

medv	age	rm	lstat	tax
24.8	22.9	6.565	9.51	293
15	100	5.757	10.11	666
15.6	98.4	5.757	17.31	437
22.6	17.2	6.333	7.34	430
15.4	100	6.485	18.85	666
48.8	91.5	8.398	5.91	264
11.7	70.6	5.565	17.16	666
21.7	17.5	5.961	9.88	305
14	93.6	6.174	24.16	437
17.2	52.3	5.898	12.67	352
21.4	45	6.286	8.94	398
19.8	97.1	6.474	12.27	384
17.1	28.4	5.706	12.43	304
7	98.3	5.414	23.97	711
9.7	97	5.036	25.68	666
18.6	18.5	5.884	7.79	411
22.2	67.8	6.456	6.73	284
30.1	80.8	6.618	7.6	307
50	89.6	5.875	8.88	666
17.4	36.8	5.594	13.09	345
15.2	91.7	6.142	18.72	307
25	93	6.101	9.81	403
24.5	42.6	5.926	13.59	391
23.3	56	5.888	13.51	276
20	100	5.344	23.09	277

4

Training dataset

Validation dataset

Training dataset

medv	age	rm	lstat	tax
24.8	22.9	6.565	9.51	293
15	100	5.757	10.11	666
15.6	98.4	5.757	17.31	437
22.6	17.2	6.333	7.34	430
15.4	100	6.485	18.85	666
48.8	91.5	8.398	5.91	264
11.7	70.6	5.565	17.16	666
21.7	17.5	5.961	9.88	305
14	93.6	6.174	24.16	437
17.2	52.3	5.898	12.67	352
21.4	45	6.286	8.94	398
19.8	97.1	6.474	12.27	384
17.1	28.4	5.706	12.43	304
7	98.3	5.414	23.97	711
9.7	97	5.036	25.68	666
18.6	18.5	5.884	7.79	411
22.2	67.8	6.456	6.73	284
30.1	80.8	6.618	7.6	307
50	89.6	5.875	8.88	666
17.4	36.8	5.594	13.09	345
15.2	91.7	6.142	18.72	307
25	93	6.101	9.81	403
24.5	42.6	5.926	13.59	391
23.3	56	5.888	13.51	276
20	100	5.344	23.09	277

5

Training dataset

Validation dataset

medv	age	rm	lstat	tax
24.8	22.9	6.565	9.51	293
15	100	5.757	10.11	666
15.6	98.4	5.757	17.31	437
22.6	17.2	6.333	7.34	430
15.4	100	6.485	18.85	666
48.8	91.5	8.398	5.91	264
11.7	70.6	5.565	17.16	666
21.7	17.5	5.961	9.88	305
14	93.6	6.174	24.16	437
17.2	52.3	5.898	12.67	352
21.4	45	6.286	8.94	398
19.8	97.1	6.474	12.27	384
17.1	28.4	5.706	12.43	304
7	98.3	5.414	23.97	711
9.7	97	5.036	25.68	666
18.6	18.5	5.884	7.79	411
22.2	67.8	6.456	6.73	284
30.1	80.8	6.618	7.6	307
50	89.6	5.875	8.88	666
17.4	36.8	5.594	13.09	345
15.2	91.7	6.142	18.72	307
25	93	6.101	9.81	403
24.5	42.6	5.926	13.59	391
23.3	56	5.888	13.51	276
20	100	5.344	23.09	277

General Machine Learning Methodology (Cross Validation)

แต่ละ **set** ทำเหมือน **validation set** ปกติ

Set 1 ให้ **RMSE(1)**
Set 2 ให้ **RMSE(2)**
Set 3 ให้ **RMSE(3)**
Set 4 ให้ **RMSE(4)**
Set 5 ให้ **RMSE(5)**

แล้วหา **RMSE average** ทั้งหมด **(1,2,3,4,5)**

1

Validation dataset

Training dataset

medv	age	rm	lstat	tax
24.8	22.9	6.565	9.51	293
15	100	5.757	10.11	666
15.6	98.4	5.757	17.31	437
22.6	17.2	6.333	7.34	430
15.4	100	6.485	18.85	666
48.8	91.5	8.398	5.91	264
11.7	70.6	5.565	17.16	666
21.7	17.5	5.961	9.88	305
14	93.6	6.174	24.16	437
17.2	52.3	5.898	12.67	352
21.4	45	6.286	8.94	398
19.8	97.1	6.474	12.27	384
17.1	28.4	5.706	12.43	304
7	98.3	5.414	23.97	711
9.7	97	5.036	25.68	666
18.6	18.5	5.884	7.79	411
22.2	67.8	6.456	6.73	284
30.1	80.8	6.618	7.6	307
50	89.6	5.875	8.88	666
17.4	36.8	5.594	13.09	345
15.2	91.7	6.142	18.72	307
25	93	6.101	9.81	403
24.5	42.6	5.926	13.59	391
23.3	56	5.888	13.51	276
20	100	5.344	23.09	277

2

Training dataset

Validation dataset

Training dataset

medv	age	rm	lstat	tax
24.8	22.9	6.565	9.51	293
15	100	5.757	10.11	666
15.6	98.4	5.757	17.31	437
22.6	17.2	6.333	7.34	430
15.4	100	6.485	18.85	666
48.8	91.5	8.398	5.91	264
11.7	70.6	5.565	17.16	666
21.7	17.5	5.961	9.88	305
14	93.6	6.174	24.16	437
17.2	52.3	5.898	12.67	352
21.4	45	6.286	8.94	398
19.8	97.1	6.474	12.27	384
17.1	28.4	5.706	12.43	304
7	98.3	5.414	23.97	711
9.7	97	5.036	25.68	666
18.6	18.5	5.884	7.79	411
22.2	67.8	6.456	6.73	284
30.1	80.8	6.618	7.6	307
50	89.6	5.875	8.88	666
17.4	36.8	5.594	13.09	345
15.2	91.7	6.142	18.72	307
25	93	6.101	9.81	403
24.5	42.6	5.926	13.59	391
23.3	56	5.888	13.51	276
20	100	5.344	23.09	277

3

Training dataset

Validation dataset

Training dataset

medv	age	rm	lstat	tax
24.8	22.9	6.565	9.51	293
15	100	5.757	10.11	666
15.6	98.4	5.757	17.31	437
22.6	17.2	6.333	7.34	430
15.4	100	6.485	18.85	666
48.8	91.5	8.398	5.91	264
11.7	70.6	5.565	17.16	666
21.7	17.5	5.961	9.88	305
14	93.6	6.174	24.16	437
17.2	52.3	5.898	12.67	352
21.4	45	6.286	8.94	398
19.8	97.1	6.474	12.27	384
17.1	28.4	5.706	12.43	304
7	98.3	5.414	23.97	711
9.7	97	5.036	25.68	666
18.6	18.5	5.884	7.79	411
22.2	67.8	6.456	6.73	284
30.1	80.8	6.618	7.6	307
50	89.6	5.875	8.88	666
17.4	36.8	5.594	13.09	345
15.2	91.7	6.142	18.72	307
25	93	6.101	9.81	403
24.5	42.6	5.926	13.59	391
23.3	56	5.888	13.51	276
20	100	5.344	23.09	277

4

Training dataset

Validation dataset

Training dataset

medv	age	rm	lstat	tax
24.8	22.9	6.565	9.51	293
15	100	5.757	10.11	666
15.6	98.4	5.757	17.31	437
22.6	17.2	6.333	7.34	430
15.4	100	6.485	18.85	666
48.8	91.5	8.398	5.91	264
11.7	70.6	5.565	17.16	666
21.7	17.5	5.961	9.88	305
14	93.6	6.174	24.16	437
17.2	52.3	5.898	12.67	352
21.4	45	6.286	8.94	398
19.8	97.1	6.474	12.27	384
17.1	28.4	5.706	12.43	304
7	98.3	5.414	23.97	711
9.7	97	5.036	25.68	666
18.6	18.5	5.884	7.79	411
22.2	67.8	6.456	6.73	284
30.1	80.8	6.618	7.6	307
50	89.6	5.875	8.88	666
17.4	36.8	5.594	13.09	345
15.2	91.7	6.142	18.72	307
25	93	6.101	9.81	403
24.5	42.6	5.926	13.59	391
23.3	56	5.888	13.51	276
20	100	5.344	23.09	277

5

Training dataset

Validation dataset

medv	age	rm	lstat	tax
24.8	22.9	6.565	9.51	293
15	100	5.757	10.11	666
15.6	98.4	5.757	17.31	437
22.6	17.2	6.333	7.34	430
15.4	100	6.485	18.85	666
48.8	91.5	8.398	5.91	264
11.7	70.6	5.565	17.16	666
21.7	17.5	5.961	9.88	305
14	93.6	6.174	24.16	437
17.2	52.3	5.898	12.67	352
21.4	45	6.286	8.94	398
19.8	97.1	6.474	12.27	384
17.1	28.4	5.706	12.43	304
7	98.3	5.414	23.97	711
9.7	97	5.036	25.68	666
18.6	18.5	5.884	7.79	411
22.2	67.8	6.456	6.73	284
30.1	80.8	6.618	7.6	307
50	89.6	5.875	8.88	666
17.4	36.8	5.594	13.09	345
15.2	91.7	6.142	18.72	307
25	93	6.101	9.81	403
24.5	42.6	5.926	13.59	391
23.3	56	5.888	13.51	276
20	100	5.344	23.09	277

Data Split

Validation Set

Hyper Parameter Tuning

Prediction & Evaluation

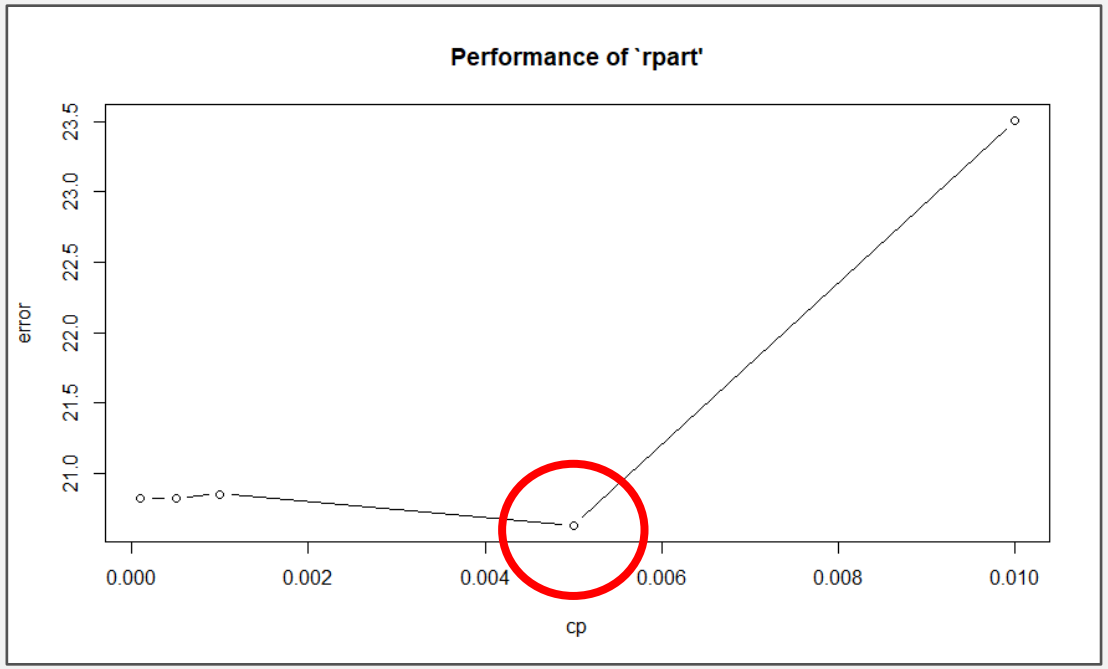
Optimization

General Machine Learning Methodology (Tuning)

- Tuning คือการหา ค่าของ **hyperparameter** ที่ทำให้ error มีขนาดเล็กที่สุด
- โดยเราจะต้อง**ลูป hyperparameter** ไปเรื่อยๆ เพื่อหาค่าที่ดีที่สุด

ยกตัวอย่างในรูปภาพ

- เรา วนลูป hyperparameter **cp** (c(0.0001,0.0005,0.001,0.005,0.01))
- เพื่อหาค่า cp ที่ทำให้ค่า error ที่น้อยที่สุด (ในที่นี้ **0.005)**



General Machine Learning Methodology (Tuning R Code)

```
library(rpart)
library(e1071)
tc <- tune.control(cross = 5)
OptModel=tune(rpart, medv~.
                  , data=train_data
                  , ranges=list(cp = c(0.0001,0.0005,0.001,0.005,0.01))
                  , tunecontrol = tc )
plot(OptModel)
OptModel

Parameter tuning of ‘rpart’:

- sampling method: 5-fold cross validation

- best parameters:
    cp
 0.005

- best performance: 20.63245
```

General Machine Learning Methodology (Tuning R Code Caret)

```
library(caret)
library(rpart)
set.seed(888)
fitControl <- trainControl(
     method = "repeatedcv",
     number = 5,
     repeats = 5)
treegrid <-  expand.grid(
     cp = c(0.0001,0.0005,0.001,0.005,0.01))

rpart_caret = train(medv~. , data = train_data , method = "rpart", trControl = fitControl ,tuneGrid = treegrid)
rpart_caret
CART

356 samples
 13 predictor

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 5 times)
Summary of sample sizes: 284, 284, 285, 287, 284, 286, ...
Resampling results across tuning parameters:
  cp     RMSE      Rsquared   MAE
  1e-04  4.720016  0.7314235  3.205382
  5e-04  4.720215  0.7313802  3.207939
  1e-03  4.719406  0.7311388  3.208152
  5e-03  4.785556  0.7221592  3.246141
  1e-02  4.877414  0.7120067  3.412466
RMSE was used to select the optimal model using the smallest value.
The final value used for the model was cp = 0.001.
```

Data Split

Validation Set

Hyper Parameter Tuning

Prediction & Evaluation

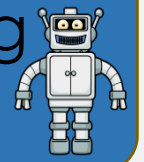
Optimization

General Machine Learning Methodology (Prediction)

New Datapoint

ข้อมูลใหม่ แต่ต้องมี Columns เหมือน Feature ตอน Train model

Machine Learning Model

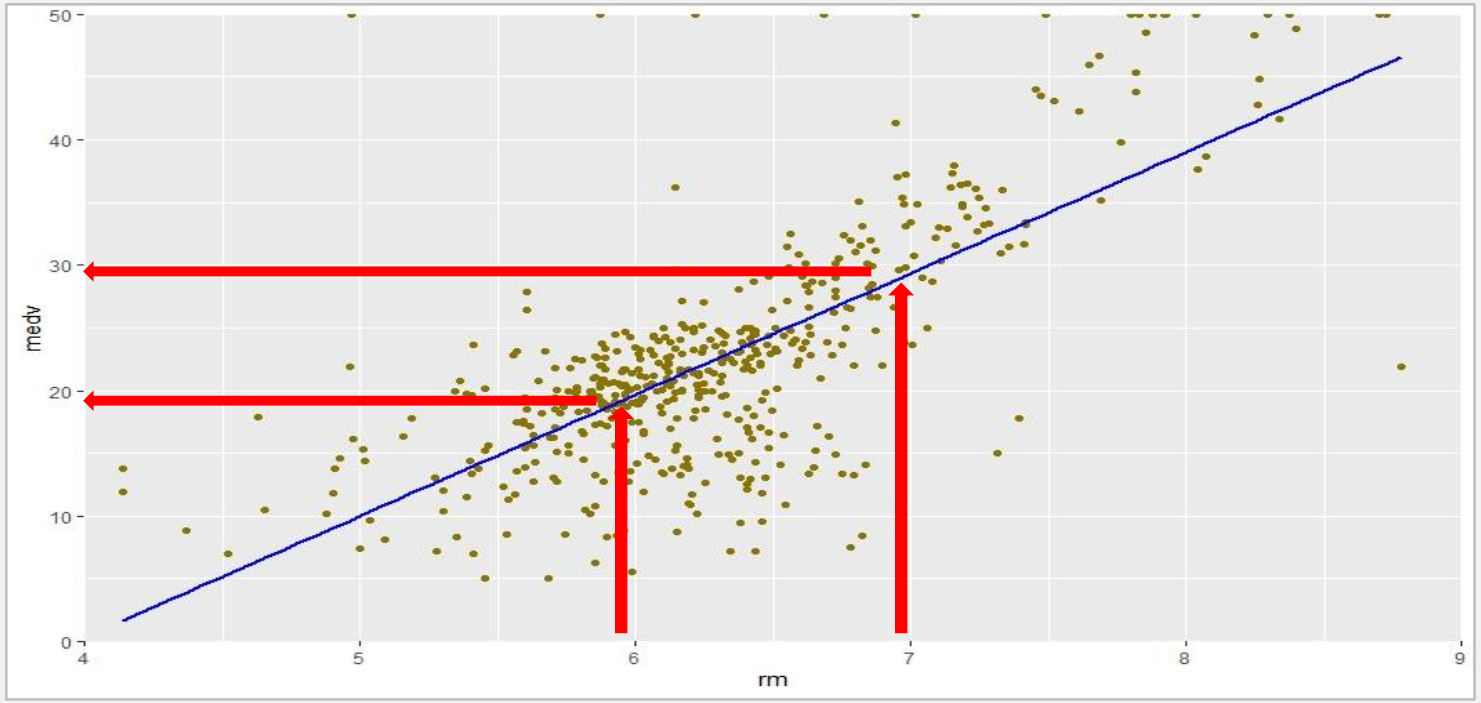


Y_predicted

General Machine Learning Methodology (Prediction)

ข้อมูลใหม่ แต่ต้องมี Columns เหมือน
Feature ตอน Train model

New Datapoint → Machine Learning Model → Y_predicted




```
linear_model <- lm(medv ~ rm , data = BostonHousing)
predict(linear_model , newdata = data.frame(rm = c(6,7)))
    1          2
19.94203    29.04414
```

General Machine Learning Methodology (Evaluation)

Predict (y_hat) vs Actual (y)

- ในการวัดผลลัพธ์ของ linear regression model จากของ test data เพื่อที่จะเป็น criteria ในการเลือก model ว่าอันไหนดีสุด

popular metrics (วัดความแม่นยำของ model)

- 1. Root Mean Squared Error:

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

- 2. Mean Absolute Error:

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}_i|$$

- 3. R_Squared

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$$

```
library(dplyr)
lm_model <- lm(medv ~ rm , data = train_data)

test_data['pred'] <- predict(lm_model , newdata = test_data)

## RMSE
RMSE(test_data$pred , test_data$medv)
[1] 5.535709

## MAE
MAE(test_data$pred , test_data$medv)
[1] 3.871203

## Rsq
(test_data %>%
        mutate(actual_y = medv) %>%
        mutate(pred_y = pred) %>%
        mutate(mean_y = mean(actual_y)) %>%
        mutate(error_squared = (pred_y - actual_y)^2) %>%
        mutate(total_squared = (actual_y - mean_y)^2) %>%
        summarize(SSE = sum(error_squared) , SST = sum(total_squared)) %>%
        summarize(R_sqaured = 1 - (SSE/SST) )
       )[1,1]
[1] 0.4418793
```

Data Split

Validation Set

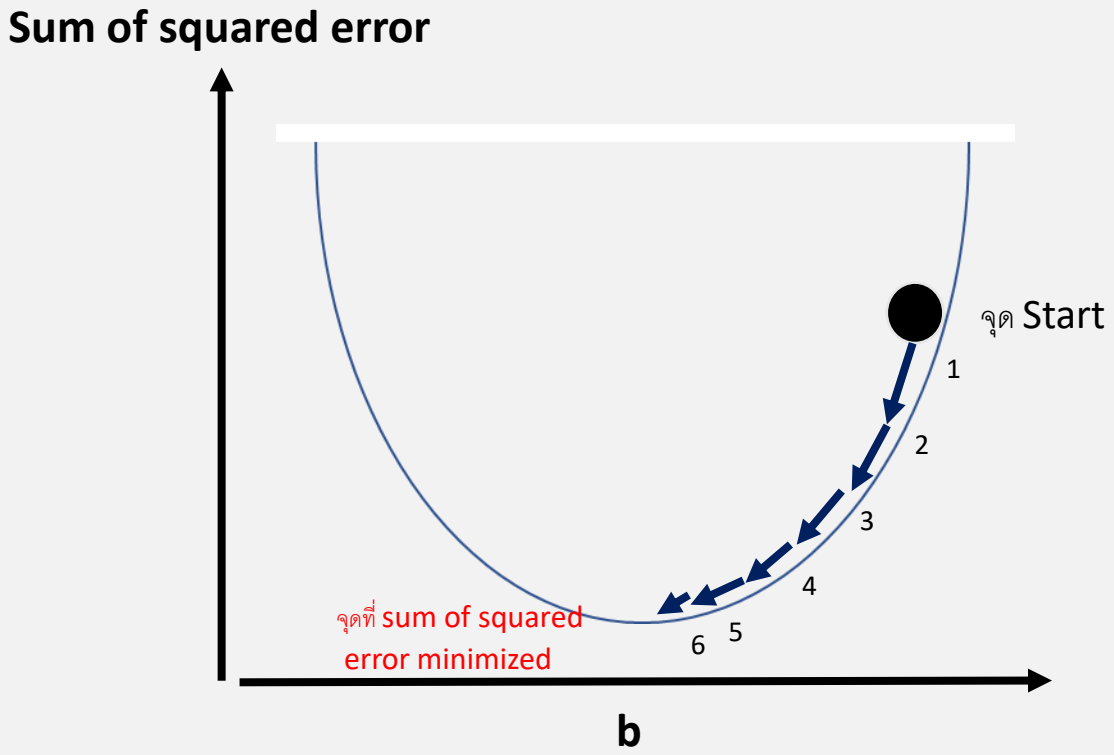
Hyper Parameter Tuning

Prediction & Evaluation

Optimization

General Machine Learning Methodology (Optimization)

- Special case ในการหา b0 b1 ของ linear regression (นอกจากใช้ normal equation video prove b0 b1 แล้ว)
  ยังมีวิธีอื่นในการหา b0 b1 ได้อีก -> optimization
- Recall หา weight (ใน linear regression คือ b0 b1) ที่ minimize cost ของ model (ในทีนี้คือ sum of squared error)
- Optimization method : **gradient descent**



1. เริ่มที่จุด start
2. หา derivative dSSE/db
3. เดินไปตาม direction ของ derivative
4. ย้อนกลับไป step 2
5. ทำไปเรื่อยๆจนถึงจุด minimum sum of squared
6. เสร็จแล้วจะได้ b0 b1 ไปทำ model สมการ

General Machine Learning Methodology (Recap)

- Regression modeling คือการสร้าง model เพื่อนำมา **predict** ข้อมูลที่เป็น **จำนวนตัวเลข** (Quantitative variable)
- เราเริ่มต้นที่ Linear regression : คือการหา **สมการเส้นตรงที่ fit ข้อมูลได้ดีที่สุด**
- Simple linear regression : คือ model ที่มี Feature **1 ตัว**
- สมการเส้นตรงหาจาก estimated b0 b1 จากข้อมูล โดยเราอยากข้อมูลเรา fit data ให้ได้เยอะที่สุด หรือ ให้เส้นตรงที่มี **sum of squared error** น้อยที่สุด
- เราต้องแบ่งข้อมูลเป็น **Training set** (เพื่อ train model) **Validation Set** (Optional), **Test set** (เพื่อ evaluated model จากค่า prediction) แล้วนำมาทำ evaluation (**RMSE , R squared)**

R Session !!!!

Interpreting Result

Interpreting result (overall model)

- อธิบายผลลัพธ์จาก model เบื้องต้น (overall model)
- R squared : model สามารถ captured การกระจายของ Y ได้มากน้อยแค่ไหน มีค่าอยู่ระหว่าง 0 - 1* ถ้า captured ได้มากค่าจะเข้าใกล้ 1 ถ้า capture ได้น้อยค่าจะเข้าใกล้ 0
- Adjusted R squared : เหมือน R squared แต่จะมีการ adjusted ให้ค่าน้อยลงถ้า เพิ่ม feature variable (X) ใหม่เข้ามาใน model แล้ว การ capture การกระจายของ Y ไม่เพิ่มขึ้น
- P-value of F test : โดยเบื้องต้น ถ้าน้อยกว่า 0.01 , 0.05 ,0.1 จะหมายถึง ทั้ง model จะมีผลสำคัญกับ Y (regression model is statistically significant at 99% , 95% , 90% level of confidence) ถ้าไม่น้อยกว่าจะหมายถึง ไม่ statistically significant

*r squared สามารถเป็นลบได้ถ้า model predict ออกมาแย่กว่า mean ของ data เช่น Trend เป็นขาขึ้น แต่ model ลากเส้นเป็นขาลงโดยมี intercept เริ่มต้นที่เดียวกัน

Interpreting result (overall model)

- อธิบายผลลัพธ์จาก model เบื้องต้น (overall model)
- R squared : model สามารถ captured การกระจายของ Y ได้มากน้อยแค่ไหน มีค่าอยู่ระหว่าง 0 - 1* ถ้า captured ได้มากค่าจะเข้าใกล้ 1 ถ้า capture ได้น้อยค่าจะเข้าใกล้ 0
- Adjusted R squared : เหมือน R squared แต่จะมีการ adjusted ให้ค่าน้อยลงถ้า เพิ่ม feature variable (X) ใหม่เข้ามาใน model แล้ว การ capture การกระจายของ Y ไม่เพิ่มขึ้น
- P-value of F test : โดยเบื้องต้น ถ้าน้อยกว่า 0.01 , 0.05 ,0.1 จะหมายถึง ทั้ง model จะมีผลสำคัญกับ Y (regression model is statistically significant at 99% , 95% , 90% level of confidence) ถ้าไม่น้อยกว่าจะหมายถึง ไม่ statistically significant

```
> summary(lm_model)

Call:
lm(formula = medv ~ rm, data = train_data)

Residuals:
    Min      1Q  Median      3Q     Max 
-22.561  -2.764   0.190   3.101  39.215 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -33.1443     3.1520  -10.52   <2e-16 ***
rm            8.8388     0.4983   17.74   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.859 on 405 degrees of freedom
Multiple R-squared:  0.4372,	Adjusted R-squared:  0.4358 
F-statistic: 314.6 on 1 and 405 DF,  p-value: < 2.2e-16
```

Adjusted R squared = 0.437

R squared = 0.437

P-value of F test = 0.00000000000000022

Interpreting result (Individual result)

- อธิบายผลลัพธ์จาก model เบื้องต้น (รายตัว)
- b1 : X เพิ่มขึ้น 1 หน่วย Y เพิ่มขึ้น b1 หน่วย
- b0 : ถ้า X = 0 , Y จะมีค่า = b0 หน่วย
- P value (b1) : ถ้าน้อยกว่า 0.01 , 0.05 ,0.1 จะหมายถึง X มีผลสำคัญกับ Y (X is statistically significant at 99% , 95% , 90% level of confidence)
- P value (b0) : ถ้าน้อยกว่า 0.01 , 0.05 ,0.1 จะหมายถึง intercept มีผลสำคัญกับ Y (intercept is statistically significant at 99% , 95% , 90% level of confidence)
- ถ้าไม่น้อยกว่าจะหมายถึง ไม่ statistically significant

Interpreting result (Individual result)

- อธิบายผลลัพธ์จาก model เบื้องต้น (รายตัว)
- b1 : X เพิ่มขึ้น 1 หน่วย Y เพิ่มขึ้น b1 หน่วย
- b0 : ถ้า X = 0 , Y จะมีค่า = b0 หน่วย
- P value (b1) : ถ้าน้อยกว่า 0.01 , 0.05 ,0.1 จะหมายถึง X มีผลสำคัญกับ Y (X is statistically significant at 99% , 95% , 90% level of confidence)
- P value (b0) : ถ้าน้อยกว่า 0.01 , 0.05 ,0.1 จะหมายถึง intercept มีผลสำคัญกับ Y (intercept is statistically significant at 99% , 95% , 90% level of confidence)
- ถ้าไม่น้อยกว่าจะหมายถึง ไม่ statistically significant

```
> summary(lm_model)

Call:
lm(formula = medv ~ rm, data = train_data)

Residuals:
    Min      1Q  Median      3Q     Max 
-22.561  -2.764   0.190   3.101  39.215 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) -33.1443     3.1520  -10.52   <2e-16 ***
rm            8.8388     0.4983   17.74   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.859 on 405 degrees of freedom
Multiple R-squared:  0.4372,	Adjusted R-squared:  0.4358 
F-statistic: 314.6 on 1 and 405 DF,  p-value: < 2.2e-16
```

ถ้า **rm = 0 , medv** จะเท่ากับ **-33.1443**

ถ้า **Rm** เพิ่ม 1 หน่วย, **medv** จะเพิ่ม 8.8388 หน่วย

P-value of b0 = 0.0000000000000002

P-value of b1 = 0.0000000000000002

Multiple Linear Regression

Linear regression theory (Equation)

Simple Linear Regression

$$y = b_0 + b_1 * x_1$$

Multiple Linear Regression

$$y = b_0 + b_1 * x_1 + b_2 * x_2 + \ldots + b_n * x_n$$

Linear regression theory (Multiple Linear Regression)

Multiple Linear Regression

$$y = b_0 + b_1 * x_1 + b_2 * x_2 + \ldots + b_n * x_n$$

medv	rm	age	dis	ptratio	b	lstat
24	6.575	65.2	4.09	15.3	396.9	4.98
21.6	6.421	78.9	4.9671	17.8	396.9	9.14
34.7	7.185	61.1	4.9671	17.8	392.83	4.03
33.4	6.998	45.8	6.0622	18.7	394.63	2.94
36.2	7.147	54.2	6.0622	18.7	396.9	5.33
28.7	6.43	58.7	6.0622	18.7	394.12	5.21
22.9	6.012	66.6	5.5605	15.2	395.6	12.43
27.1	6.172	96.1	5.9505	15.2	396.9	19.15
16.5	5.631	100	6.0821	15.2	386.63	29.93
18.9	6.004	85.9	6.5921	15.2	386.71	17.1
15	6.377	94.3	6.3467	15.2	392.52	20.45
18.9	6.009	82.9	6.2267	15.2	396.9	13.27
21.7	5.889	39	5.4509	15.2	390.5	15.71
20.4	5.949	61.8	4.7075	21	396.9	8.26
18.2	6.096	84.5	4.4619	21	380.02	10.26
...	...	...	...	...	...	...
...	...	...	...	...	...	...

Medv = b0 + b1*Rm +b2*age + b3*dis + b4*ptratio +b5*b + b6*lstat

ราคาบ้านโดยเฉลี่ย (1,000 USD) | Y | Respond variable | Outcome variable

(rm , age, dis , ptratio , b , lstat) | X | Factor | Features | predictor variable

Multiple Linear Regression

$$y = b_0 + b_1 * x_1 + b_2 * x_2 + \ldots + b_n * x_n$$

โดยการที่เพิ่มตัวแปรเข้ามาจะมีดีข้อเสียดังนี้

ข้อดี :

1. Model มีโอกาสที่จะ Predict ได้ดีขึ้นมาก เพราะ complex มากขึ้น
2. ลด ปัญหา Bias (ลด under fitting)

ข้อเสีย :

1. Model มักจะใช้ เวลา Train นานขึ้น
2. เกิดปัญหา variance (เพิ่ม over fitting)
3. เนื่องจาก Model Complex มากขึ้นทำให้การอธิบาย ผลเปลี่ยนไป

Linear regression theory (Multiple Linear Regression)

```
linear_model_three <-  lm(medv ~ rm + age +dis  , data =
BostonHousing)
summary(linear_model_three)

Call:
lm(formula = medv ~ rm + age + dis, data = BostonHousing)

Residuals:
    Min      1Q  Median      3Q     Max
-21.179  -2.808  -0.215   2.166  40.505

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -21.87279    3.17796  -6.883 1.76e-11 ***
rm            8.44063    0.41047  20.563  < 2e-16 ***
age          -0.09942    0.01511  -6.582 1.17e-10 ***
dis          -0.48038    0.20028  -2.399   0.0168 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.286 on 502 degrees of freedom
Multiple R-squared:  0.5356,	Adjusted R-squared:  0.5328
F-statistic:   193 on 3 and 502 DF,  p-value: < 2.2e-16
```

MULTIPLE LINEAR REGRESSION
(ใช้บาง Column
[rm , age, dis])

```
linear_model_multiple <- lm(medv ~ . , data = BostonHousing)
summary(linear_model_multiple)

Call:
lm(formula = medv ~ ., data = BostonHousing)

Residuals:
    Min      1Q  Median      3Q     Max
-15.595  -2.730  -0.518   1.777  26.199

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.646e+01  5.103e+00   7.144 3.28e-12 ***
crim        -1.080e-01  3.286e-02  -3.287 0.001087 **
zn           4.642e-02  1.373e-02   3.382 0.000778 ***
indus        2.056e-02  6.150e-02   0.334 0.738288
chas1        2.687e+00  8.616e-01   3.118 0.001925 **
nox         -1.777e+01  3.820e+00  -4.651 4.25e-06 ***
rm           3.810e+00  4.179e-01   9.116  < 2e-16 ***
age          6.922e-04  1.321e-02   0.052 0.958229
dis         -1.476e+00  1.995e-01  -7.398 6.01e-13 ***
rad          3.060e-01  6.635e-02   4.613 5.07e-06 ***
tax         -1.233e-02  3.760e-03  -3.280 0.001112 **
ptratio     -9.527e-01  1.308e-01  -7.283 1.31e-12 ***
b            9.312e-03  2.686e-03   3.467 0.000573 ***
lstat       -5.248e-01  5.072e-02 -10.347  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.745 on 492 degrees of freedom
Multiple R-squared:  0.7406,	Adjusted R-squared:  0.7338
F-statistic: 108.1 on 13 and 492 DF,  p-value: < 2.2e-16
```

MULTIPLE LINEAR REGRESSION
(ใช้ทุก Column)

Linear regression theory (Multiple Linear Regression caret)

Caret Version

MULTIPLE LINEAR REGRESSION (ใช้บาง Column [rm , age, dis])

```
library(caret)
linear_model_three_caret <- train(
    form = medv ~ rm + age +dis ,
    data = BostonHousing,
    method = 'lm'
)
summary(linear_model_three_caret)
```

```
Call:
lm(formula = medv ~ rm + age + dis, data = BostonHousing)

Residuals:
    Min      1Q  Median      3Q     Max 
-21.179  -2.808  -0.215   2.166  40.505 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) -21.87279    3.17796  -6.883 1.76e-11 ***
rm            8.44063    0.41047  20.563  < 2e-16 ***
age          -0.09942    0.01511  -6.582 1.17e-10 ***
dis          -0.48038    0.20028  -2.399   0.0168 *  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 6.286 on 502 degrees of freedom
Multiple R-squared:  0.5356,	Adjusted R-squared:  0.5328 
F-statistic:   193 on 3 and 502 DF,  p-value: < 2.2e-16
```

Caret Version

MULTIPLE LINEAR REGRESSION (ใช้ทุก Column)

```
library(caret)
linear_model_multiple_caret <- train(
    form = medv ~ . ,
    data = BostonHousing,
    method = 'lm'
)
summary(linear_model_multiple_caret)
```

```
Call:
lm(formula = medv ~ ., data = BostonHousing)

Residuals:
    Min      1Q  Median      3Q     Max 
-15.595  -2.730  -0.518   1.777  26.199 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept)  3.646e+01  5.103e+00   7.144 3.28e-12 ***
crim        -1.080e-01  3.286e-02  -3.287 0.001087 ** 
zn           4.642e-02  1.373e-02   3.382 0.000778 ***
indus        2.056e-02  6.150e-02   0.334 0.738288    
chas1        2.687e+00  8.616e-01   3.118 0.001925 ** 
nox         -1.777e+01  3.820e+00  -4.651 4.25e-06 ***
rm           3.810e+00  4.179e-01   9.116  < 2e-16 ***
age          6.922e-04  1.321e-02   0.052 0.958229    
dis         -1.476e+00  1.995e-01  -7.398 6.01e-13 ***
rad          3.060e-01  6.635e-02   4.613 5.07e-06 ***
tax         -1.233e-02  3.760e-03  -3.280 0.001112 ** 
ptratio     -9.527e-01  1.308e-01  -7.283 1.31e-12 ***
b            9.312e-03  2.686e-03   3.467 0.000573 ***
lstat       -5.248e-01  5.072e-02 -10.347  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.745 on 492 degrees of freedom
Multiple R-squared:  0.7406,	Adjusted R-squared:  0.7338 
F-statistic: 108.1 on 13 and 492 DF,  p-value: < 2.2e-16
```

Interpretation

b1 : X เพิ่มขึ้น 1 หน่วย Y เพิ่มขึ้น b1 หน่วย ถ้า factor อื่นไม่เปลี่ยนแปลง (อธิบาย เหมือน b1 b2 b3....)
Ex: dis เพิ่มขึ้น 1 หน่วย Y ลดลง 0.8855 หน่วย (เนื่องจาก เครื่องหมายลบ) ถ้า factor อื่นไม่เปลี่ยนแปลง

```
> lm_mul_model <- lm(medv~  rm+ age+ dis+ ptratio +b +lstat, data = train_data)
> summary(lm_mul_model)

Call:
lm(formula = medv ~ rm + age + dis + ptratio + b + lstat, data = train_data)

Residuals:
     Min       1Q   Median       3Q      Max 
-15.0734  -2.8774  -0.7928   1.7989  27.2937 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) 21.642897   5.045390   4.290 2.25e-05 ***
rm           4.140423   0.493280   8.394 8.24e-16 ***
age         -0.028437   0.014817  -1.919   0.0557 .  
dis         -0.885583   0.180808  -4.898 1.41e-06 ***
ptratio     -0.901604   0.135408  -6.658 9.16e-11 ***
b            0.012489   0.003027   4.126 4.50e-05 ***
lstat       -0.603717   0.058442 -10.330  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5.172 on 400 degrees of freedom
Multiple R-squared:  0.6839,	Adjusted R-squared:  0.6792 
F-statistic: 144.2 on 6 and 400 DF,  p-value: < 2.2e-16
```

Linear regression theory (Categorical Input)

หาก Feature Column ไม่ใช้ ตัวเลข (numeric) แต่เป็น Category หรือ Factor เช่น เพศ Female/ Male

โดยปกติแล้ว เราต้อง convert Feature Column นั้น ให้เป็น Factor เช่น Female = 0 , Male = 1

แต่ R สามารถ Convert ให้เราได้ เวลาที่เราทำ regression

weight	gender	height
55	F	165
45	F	159
41	F	151
60	M	165
70	M	171
75	M	180

Linear Regression : Y = b0 + b1*X1 +b2*X2
Linear Regression : Weight = b0 + b1* SexM + b2*height

Intrepret :
หาก SexM = 0 [female] : Weight = b0 + b2*height
หาก SexM =1 [male] : Weight = b0 + **8.0951 +** b2*height
ง่ายๆคือ หาก เป็น male ; Weight โดยเฉลี่ยจะมี ค่ามากกว่า female อยู่ **8.0951**

```
weight <- c(55 , 45,41,60,70,75)
sex <- c('F','F','F','M','M','M')
height <- c(165,159,151,165,171,180)
weight_data <- data.frame(weight, sex,
height)
weight_data
  weight sex height
1     55   F    165
2     45   F    159
3     41   F    151
4     60   M    165
5     70   M    171
6     75   M    180
lm(weight ~ . , data = weight_data)

Call:
lm(formula = weight ~ ., data =
weight_data)

Coefficients:
(Intercept)         sexM       height
  -106.3699       8.0951       0.9687
```

Multiple Linear Regression Coefficients Prove!!! (Optional)

Nonlinear Modeling

Linear regression theory (Polynomial Regression)

Assumption ของ linear regression model คือ assume ว่าความสัมพันธ์ระหว่าง respond variable (Y) กับ predictor (X) เป็น linear

แต่ถ้าข้อมูลเป็น Non linear relationship ?

ใช้ **Polynomial regression :** เพิ่มตัวกำลังของ X เข้าไป เช่น X กำลัง 2 , X กำลัง 3 ..... (เพื่อที่จะ modeling nonlinear relationship ได้)

Simple Linear Regression

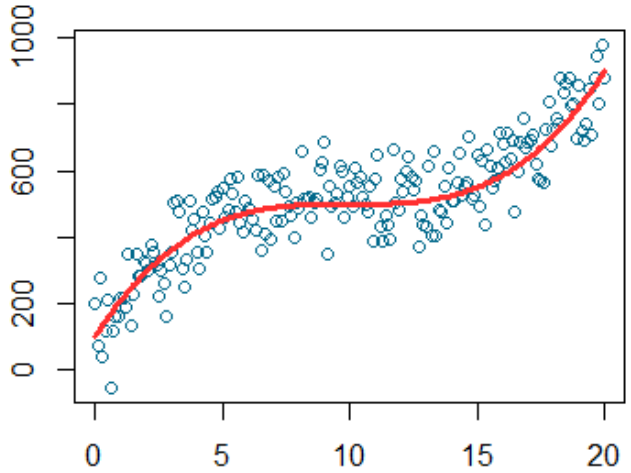
$$y = b_0 + b_1 x_1$$

Multiple Linear Regression

$$y = b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_n x_n$$

Polynomial Linear Regression

$$y = b_0 + b_1 x_1 + b_2 x_1^2 + \ldots + b_n x_1^n$$



Source: https://medium.com/analytics-vidhya/machine-learning-project-3-predict-salary-using-polynomial-regression-7024c7bace4f

Linear regression theory (Polynomial Regression)

Assumption ของ linear regression model คือ assume ว่าความสัมพันธ์ระหว่าง respond variable (Y) กับ predictor (X) เป็น linear

แต่ถ้าข้อมูลเป็น Nonlinear relationship ?

ใช้ **Polynomial regression :** เพิ่มตัวกำลังของ X เข้าไป เช่น X ยกกำลัง 2 , X ยกกำลัง 3 ....(เพื่อที่จะ modeling Nonlinear relationship ได้)

กำลัง 2 กำลัง 3 กำลัง 4

medv	lstat	lstat^2	lstat^3	lstat^4
24	4.98	24.8004	123.506	615.0598
21.6	9.14	83.5396	763.5519	6978.865
34.7	4.03	16.2409	65.45083	263.7668
33.4	2.94	8.6436	25.41218	74.71182
36.2	5.33	28.4089	151.4194	807.0656
28.7	5.21	27.1441	141.4208	736.8022
22.9	12.43	154.5049	1920.496	23871.76
27.1	19.15	366.7225	7022.736	134485.4
16.5	29.93	895.8049	26811.44	802466.4
18.9	17.1	292.41	5000.211	85503.61
15	20.45	418.2025	8552.241	174893.3
18.9	13.27	176.0929	2336.753	31008.71
21.7	15.71	246.8041	3877.292	60912.26
20.4	8.26	68.2276	563.56	4655.005
18.2	10.26	105.2676	1080.046	11081.27

Linear regression theory (Polynomial Regression)

R Code มี 2 รูปแบบ
สมมุติว่า polynomial กำลัง 4

1. ใช้ I( ) เช่น I(x ^2) + I(x ^3)+I(x^4) เพิ่มเข้าไป

```
poly_model_I <- lm(medv~ lstat+ I(lstat^2) + I(lstat^3) + I(lstat^4), data = train_data)

summary(poly_model_I)
```

2. ใช้ poly() เขียนแค่ ตัวเดียว poly(x ,4) แทน feature ทั้งหมดได้เลย

```
poly_model_poly <- lm(medv~ poly(lstat,4), data = train_data)

summary(poly_model_poly)
```

ข้อแตกต่างระหว่าง I() กับ poly()

* ใช้ I() จะได้ผลลัพธ์เหมือนเอา feature ในตารางหน้าที่แล้วมาใส่
* ใช้ poly() ผลลัพธ์เส้นจะเหมือนกัน (ใน training set) แต่ พวก coefficient ไม่เหมือนเพราะว่า poly() ทำไปเพื่อแก้ปัญหา correlation ระหว่างตัว feature ที่มีต่าง degree กัน เช่น x^2 กับ x^3 แต่ poly() จะ interpret ไม่ได้

Caret Version

R Code มี 2 รูปแบบ
สมมุติว่า polynomial กำลัง 4

1. ใช้ I( ) เช่น I(x ^2) + I(x ^3)+I(x^4) เพิ่มเข้าไป

```
library(caret)
poly_model_I_caret <- train(
   form = medv~ lstat+ I(lstat^2) + I(lstat^3) + I(lstat^4) ,
   data = train_data,
   method = 'lm'
)
summary(poly_model_I_caret)
```

2. ใช้ poly() เขียนแค่ ตัวเดียว poly(x ,4) แทน feature ทั้งหมดได้เลย

```
library(caret)
poly_model_poly_caret <- train(
   form = medv~ poly(lstat,4) ,
   data = train_data,
   method = 'lm'
)
summary(poly_model_I_caret)
```

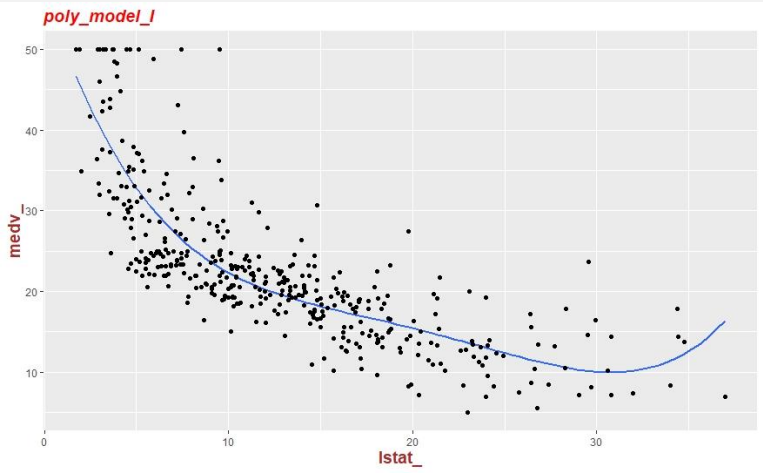
ข้อแตกต่างระหว่าง I() กับ poly()

* ใช้ I() จะได้ผลลัพธ์เหมือนเอา feature ในตารางหน้าที่แล้วมาใส่
* ใช้ poly() ผลลัพธ์เส้นจะเหมือนกัน (ใน training set) แต่ พวก coefficient ไม่เหมือนเพราะว่า poly() ทำไปเพื่อแก้ปัญหา correlation ระหว่างตัว feature ที่มีต่าง degree กัน เช่น x^2 กับ x^3 แต่ poly() จะ interpret ไม่ได้

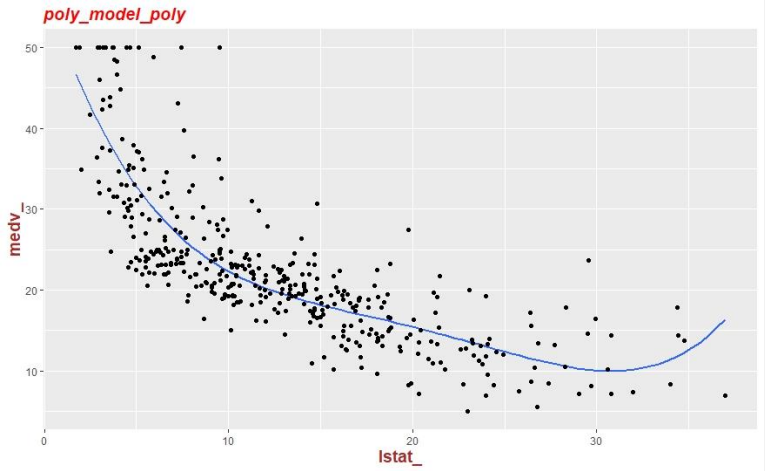
Linear regression theory (Polynomial Regression)

R Code summary

```
> poly_model_I <- lm(medv~  lstat+ I(lstat^2) + I(lstat^3) + I(lstat^4), data = train_data)
> summary(poly_model_I)

Call:
lm(formula = medv ~ lstat + I(lstat^2) + I(lstat^3) + I(lstat^4),
    data = train_data)

Residuals:
     Min       1Q   Median       3Q      Max
-13.0993  -3.1826  -0.4446   2.1827  27.0531

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  5.739e+01  2.537e+00  22.623  < 2e-16 ***
lstat       -6.995e+00  8.407e-01  -8.320 1.39e-15 ***
I(lstat^2)   4.925e-01  8.862e-02   5.558 4.99e-08 ***
I(lstat^3)  -1.635e-02  3.635e-03  -4.496 9.05e-06 ***
I(lstat^4)   1.982e-04  5.027e-05   3.943 9.49e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.325 on 402 degrees of freedom
Multiple R-squared:  0.6852,    Adjusted R-squared:  0.682
F-statistic: 218.7 on 4 and 402 DF,  p-value: < 2.2e-16
```

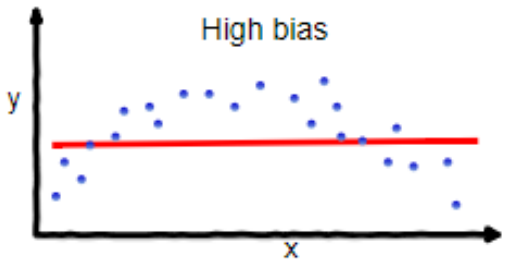



```
> poly_model_poly <- lm(medv~  poly(lstat,4), data = train_data)
> summary(poly_model_poly)

Call:
lm(formula = medv ~ poly(lstat, 4), data = train_data)

Residuals:
     Min       1Q   Median       3Q      Max
-13.0993  -3.1826  -0.4446   2.1827  27.0531

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)        22.6447     0.2639  85.796  < 2e-16 ***
poly(lstat, 4)1 -141.7639     5.3247 -26.624  < 2e-16 ***
poly(lstat, 4)2   60.4296     5.3247  11.349  < 2e-16 ***
poly(lstat, 4)3  -24.8158     5.3247  -4.661 4.30e-06 ***
poly(lstat, 4)4   20.9952     5.3247   3.943 9.49e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.325 on 402 degrees of freedom
Multiple R-squared:  0.6852,    Adjusted R-squared:  0.682
F-statistic: 218.7 on 4 and 402 DF,  p-value: < 2.2e-16
```

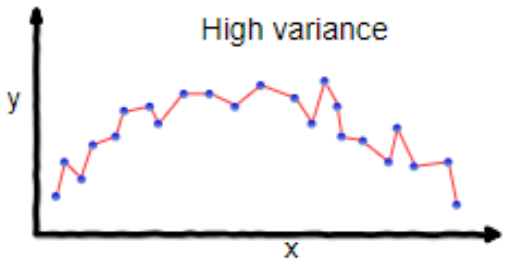


Potential problem of complex modeling

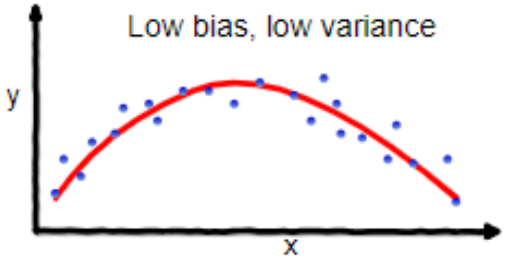
Linear regression theory (Underfit vs Overfit)



Underfitting



Overfitting

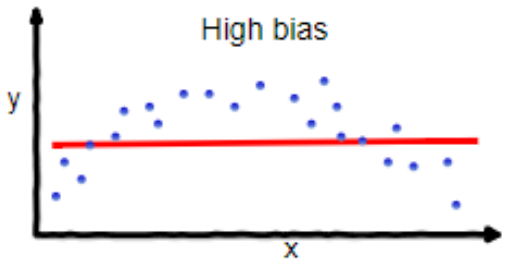


Balance

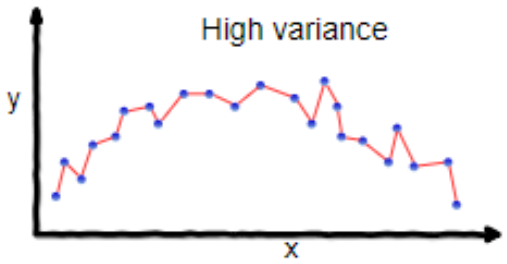
Underfitting

- โดยปกติแล้ว model ที่มี **feature น้อย**มากและไม่สำคัญจนไม่สามารถ capture การกระจายของ Y ได้ดี -> **high bias / underfitting**
- **Error** (เช่น RMSE) ใน **training** dataset จะสูง และ testing dataset จะสูงตามด้วย
- **แก้โดยการเพิ่ม Features** เข้าไปใน model ไม่ว่าจะเป็น business feature หรือ polynomial feature (หากเพิ่มแล้วไม่สามารถ reduce error ได้อาจจะต้อง transform variable เพิ่มหรือ เปลี่ยน model)

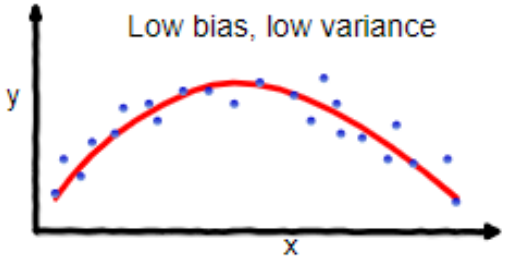
Linear regression theory (Underfit vs Overfit)



Underfitting



Overfitting

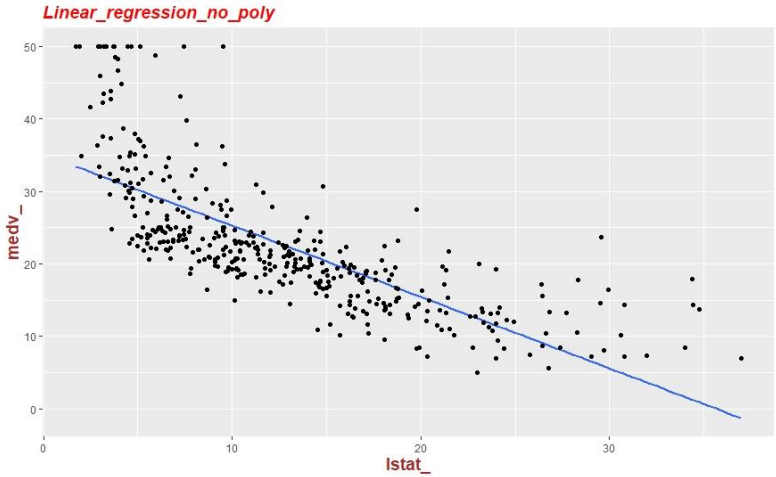


Balance

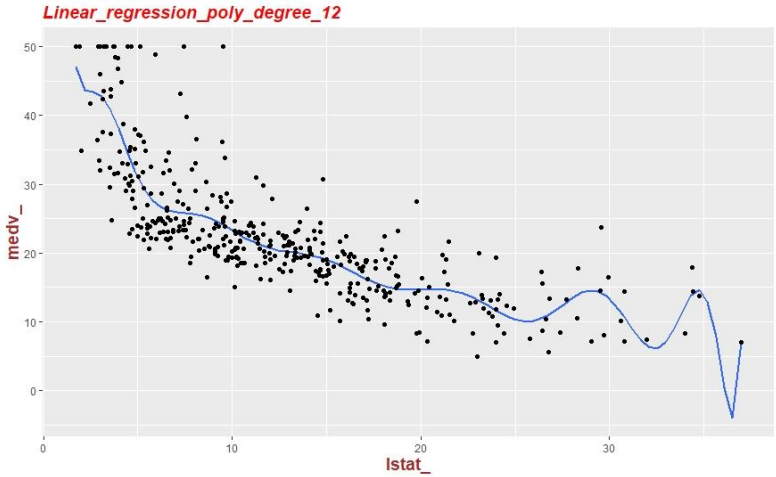
Overfitting

- โดยปกติแล้ว model ที่มี **Feature มากเกิน** capture การกระจายของ Y และ additional noise มากเกินไป -> **high variance / over fitting**
- **Error** (เช่น RMSE) ใน **training dataset จะต่ำ แต่ testing dataset จะสูง**
- **แก้โดยการลด Features ที่ไม่สำคัญออกจาก model** โดยอาจจะใช้ ridge regression / lasso regression ในการช่วยลดการ over fit

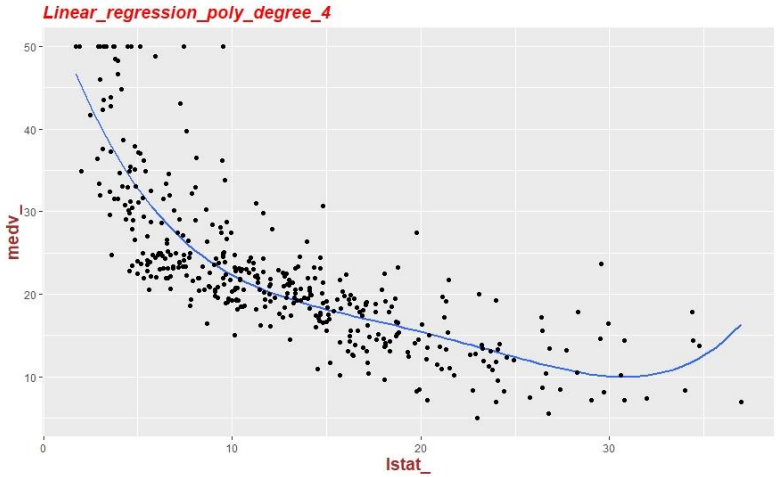
Linear regression theory (Underfit vs Overfit)



Underfitting
(Bias Problem)



Overfitting
(Variance Problem)



Balance
(Great !!!)

Linear regression theory (Underfit vs Overfit)

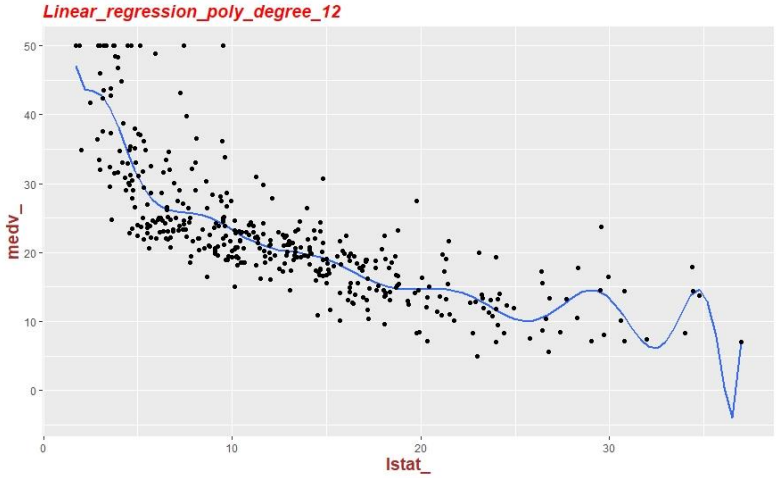
Multiple Linear Regression

$$y = b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_n x_n$$

Polynomial Linear Regression

$$y = b_0 + b_1 x_1 + b_2 x_1^2 + \ldots + b_n x_1^n$$

ถ้า Model Complex มากเกินไป



Overfitting
(Variance Problem)

Linear regression theory (Underfit vs Overfit)

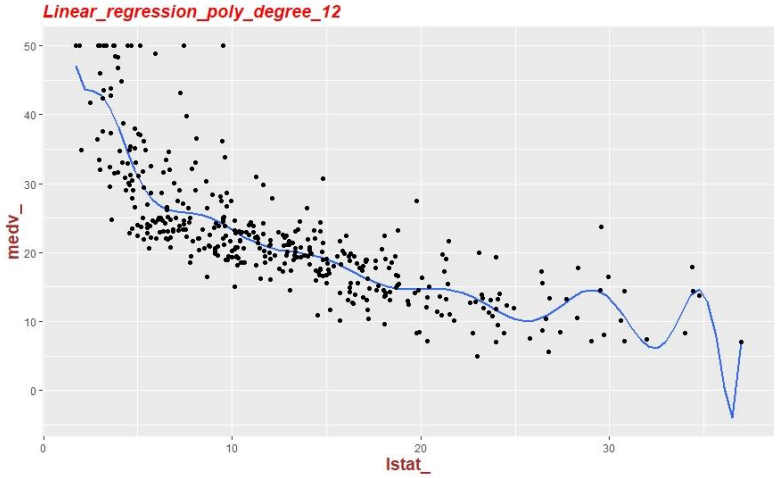
Multiple Linear Regression

$$y = b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_n x_n$$

Polynomial Linear Regression

$$y = b_0 + b_1 x_1 + b_2 x_1^2 + \ldots + b_n x_1^n$$

ถ้า Model Complex มากเกินไป



Overfitting
(Variance Problem)

Solved By

Regularization !!!

Regularization
(Ridge Regression and Lasso Regression)

Ridge regression (L2)

cost function ของ regression model เดิมคือ **sum of squared error**

Linear Regression Cost $= \sum(\hat{y}_i - y_i)$ ^2

Ridge regression ใส่ ผลรวมของ coefficient กำลัง 2 เข้าไปใน cost function

Ridge Regression Cost $= \sum(\hat{y}_i - y_i)$ ^2 + $\lambda \sum$ bj^2

- ถ้า feature ที่เพิ่มเข้ามาเยอะมาก -> ก้อนหลังเยอะขึ้น -> **total cost เราเยอะขึ้น**
- เนื่องจากเราอยาก minimize cost -> ridge regression ทำให้ค่าของ coefficient น้อยลง

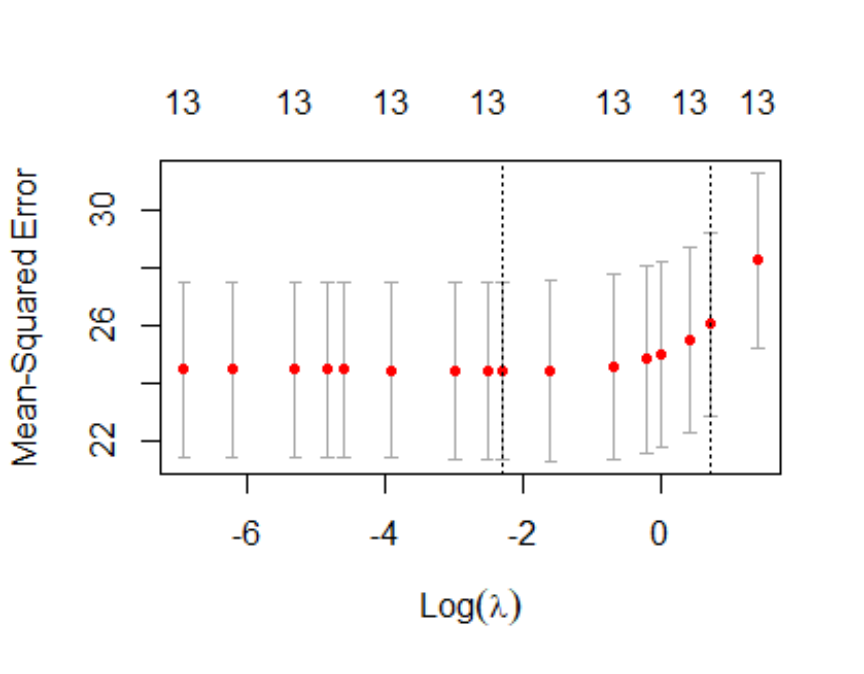
Linear regression theory (Ridge Regression R code)

```
library(dplyr)
library(glmnet)
library(caret)
y_train <- training_scaled$medv
x_train <- training_scaled[,-which(names(training_scaled) == "medv")] %>% data.matrix()

y_test <- testing_scaled$medv
x_test <- testing_scaled[,-which(names(testing_scaled) == "medv")] %>% data.matrix()

set.seed(1689)
ridge_model <- glmnet(x_train , y_train ,alpha = 0, lambda = 0.02)
y_pred <-  predict(ridge_model , newx = x_test)
RMSE(y_pred , y_test)
[1] 4.458042
coef(ridge_model)
14 x 1 sparse Matrix of class "dgCMatrix"
                      s0
(Intercept) 20.1396696545
crim        -0.9381584167
zn           1.2235823383
indus       -0.0004088653
chas         2.4072184225
nox         -2.1924272083
rm           2.4561635154
age          0.3443239237
dis         -3.2538683909
rad          2.9608994110
tax         -2.0129017870
ptratio     -2.2205140380
b            0.9211151788
lstat       -4.2401190207
```

glmnet data จะต้องอยู่ในรูป matrix !!!

Linear regression theory (Ridge Regression R code)

```
set.seed(1689)
lambda_set = c(0.001,0.002,0.005,0.008,0.01,0.02,0.05,0.08,0.1,0.2,0.5,0.8,1,1.5,2,4)
cv_ridge <- cv.glmnet(x_train, y_train ,nfolds = 5, alpha = 0, lambda = lambda_set)
plot(cv_ridge)
best_lambda <- cv_ridge$lambda.min
best_lambda
[1] 0.1
```



Linear regression theory (Ridge Regression caret)

Caret Version

```
library(caret)
library(glmnet)
lasso_model <- train(
                medv ~ .,
                        data = BostonHousing,
                        method = "glmnet",
                        tuneGrid = expand.grid(
                                alpha = 0,
                                lambda = 0.2
                                                )
                    )
lasso_model
```

หากต้องการ tune hyperparameter
alpha หรือ lambda ให้ใส่ value เข้าไปเป็น vector

เช่น
alpha = c(0,0.1,0.2,0.5,0.8,1)
lambda = c(0,0.01,0.02,0.05,0.1,0.5,1,2)

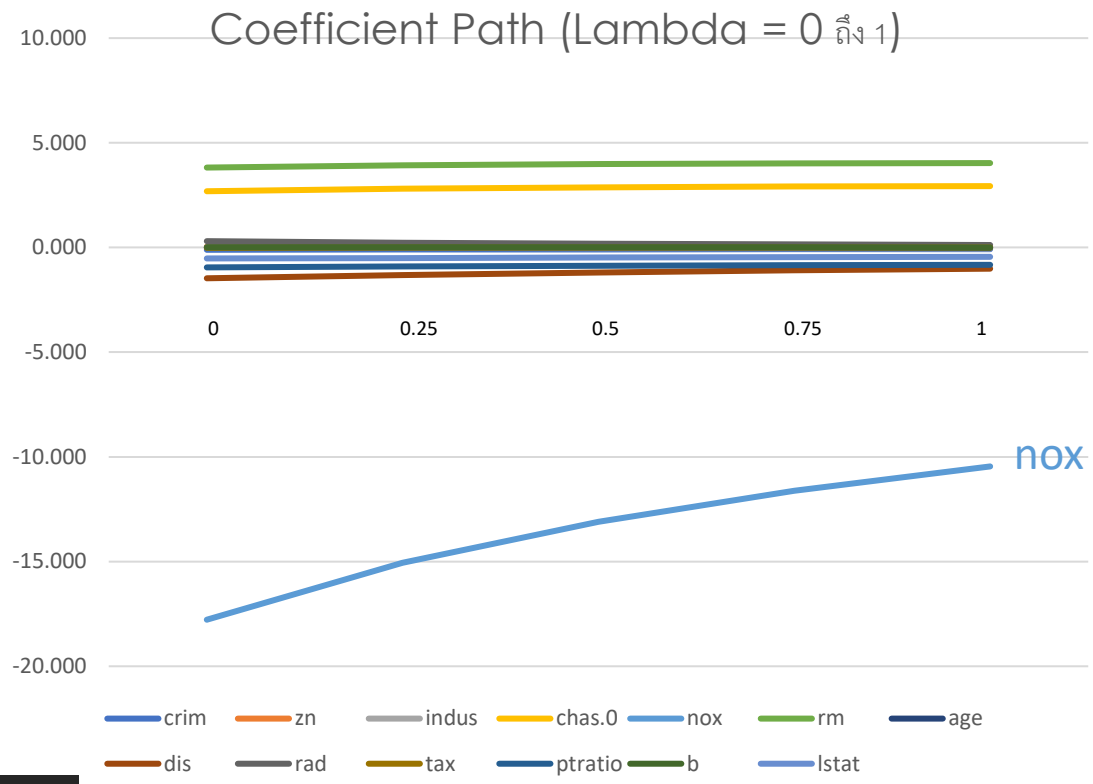
glmnet model
alpha (ridge ให้ set = 0, lasso ให้ set = 1)
Lambda (normal linear regression ให้ set = 0 , ถ้าจะ penalized parameter ให้ set >0 ยิ่งเยอะยิ่ง penalized แรง)

Linear regression theory (Ridge Regression)

Lambda

Ridge Regression Cost $= \sum(\hat{y}_i - y_i)^2 + \lambda \sum bj^2$

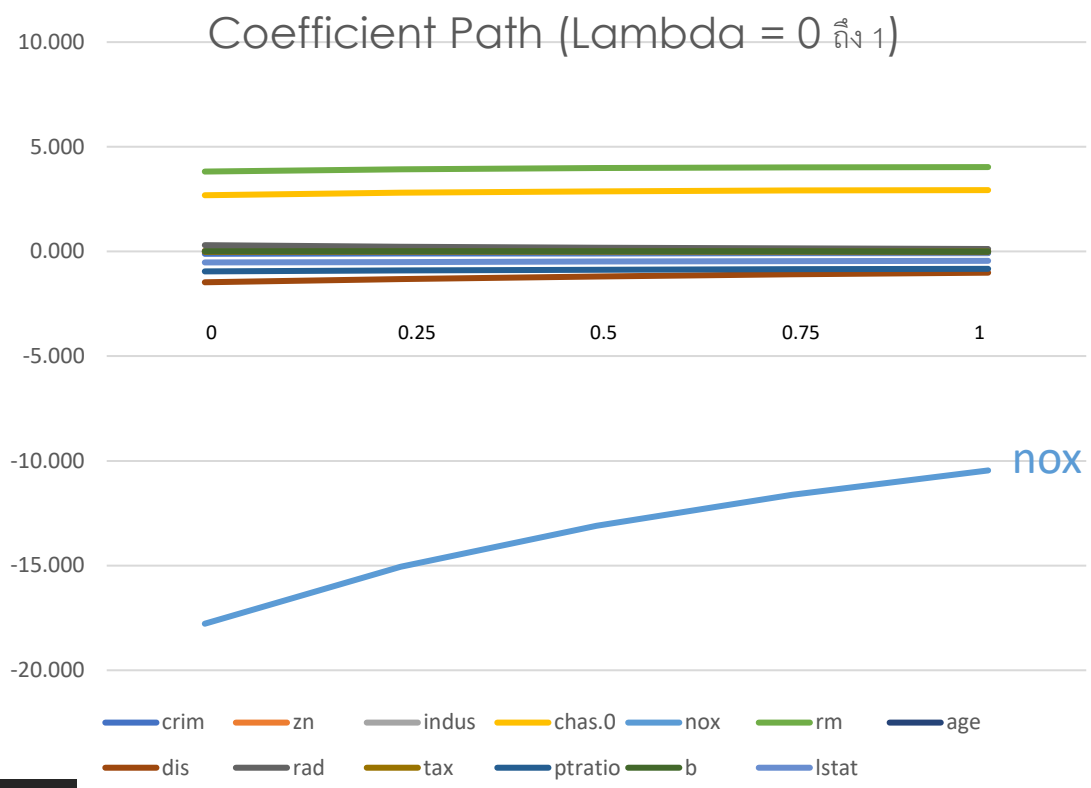
coef/lambda	0	0.25	0.5	0.75	1
crim	-0.108	-0.098	-0.091	-0.087	-0.083
zn	0.046	0.040	0.035	0.032	0.029
indus	0.020	-0.014	-0.031	-0.041	-0.047
chas.0	2.691	2.816	2.878	2.910	2.925
nox	**-17.771**	**-15.052**	**-13.101**	**-11.613**	**-10.459**
rm	3.813	3.929	3.991	4.020	4.025
age	0.001	-0.002	-0.003	-0.004	-0.005
dis	-1.477	-1.323	-1.197	-1.093	-1.007
rad	0.304	0.224	0.177	0.146	0.124
tax	-0.012	-0.009	-0.007	-0.005	-0.005
ptratio	-0.952	-0.908	-0.875	-0.849	-0.826
b	0.009	0.009	0.009	0.009	0.009
lstat	-0.524	-0.502	-0.483	-0.467	-0.453




```
summary(BostonHousing)
      crim                zn             indus       chas        nox               rm             age              dis
 Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   0:471   Min.   :0.3850   Min.   :3.561   Min.   :  2.90   Min.   : 1.130
 1st Qu.: 0.08204   1st Qu.:  0.00   1st Qu.: 5.19   1: 35   1st Qu.:0.4490   1st Qu.:5.886   1st Qu.: 45.02   1st Qu.: 2.100
 Median : 0.25651   Median :  0.00   Median : 9.69           Median :0.5380   Median :6.208   Median : 77.50   Median : 3.207
 Mean   : 3.61352   Mean   : 11.36   Mean   :11.14           Mean   :0.5547   Mean   :6.285   Mean   : 68.57   Mean   : 3.795
 3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10           3rd Qu.:0.6240   3rd Qu.:6.623   3rd Qu.: 94.08   3rd Qu.: 5.188
 Max.   :88.97620   Max.   :100.00   Max.   :27.74           Max.   :0.8710   Max.   :8.780   Max.   :100.00   Max.   :12.127
      rad              tax           ptratio            b               lstat            medv
 Min.   : 1.000   Min.   :187.0   Min.   :12.60   Min.   :  0.32   Min.   : 1.73   Min.   : 5.00
 1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40   1st Qu.:375.38   1st Qu.: 6.95   1st Qu.:17.02
 Median : 5.000   Median :330.0   Median :19.05   Median :391.44   Median :11.36   Median :21.20
 Mean   : 9.549   Mean   :408.2   Mean   :18.46   Mean   :356.67   Mean   :12.65   Mean   :22.53
 3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:396.23   3rd Qu.:16.95   3rd Qu.:25.00
 Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :396.90   Max.   :37.97   Max.   :50.00
```

Linear regression theory (Ridge Regression)

coef/lambda	0	0.25	0.5	0.75	1
crim	-0.108	-0.098	-0.091	-0.087	-0.083
zn	0.046	0.040	0.035	0.032	0.029
indus	0.020	-0.014	-0.031	-0.041	-0.047
chas.0	2.691	2.816	2.878	2.910	2.925
nox	**-17.771**	**-15.052**	**-13.101**	**-11.613**	**-10.459**
rm	3.813	3.929	3.991	4.020	4.025
age	0.001	-0.002	-0.003	-0.004	-0.005
dis	-1.477	-1.323	-1.197	-1.093	-1.007
rad	0.304	0.224	0.177	0.146	0.124
tax	-0.012	-0.009	-0.007	-0.005	-0.005
ptratio	-0.952	-0.908	-0.875	-0.849	-0.826
b	0.009	0.009	0.009	0.009	0.009
lstat	-0.524	-0.502	-0.483	-0.467	-0.453



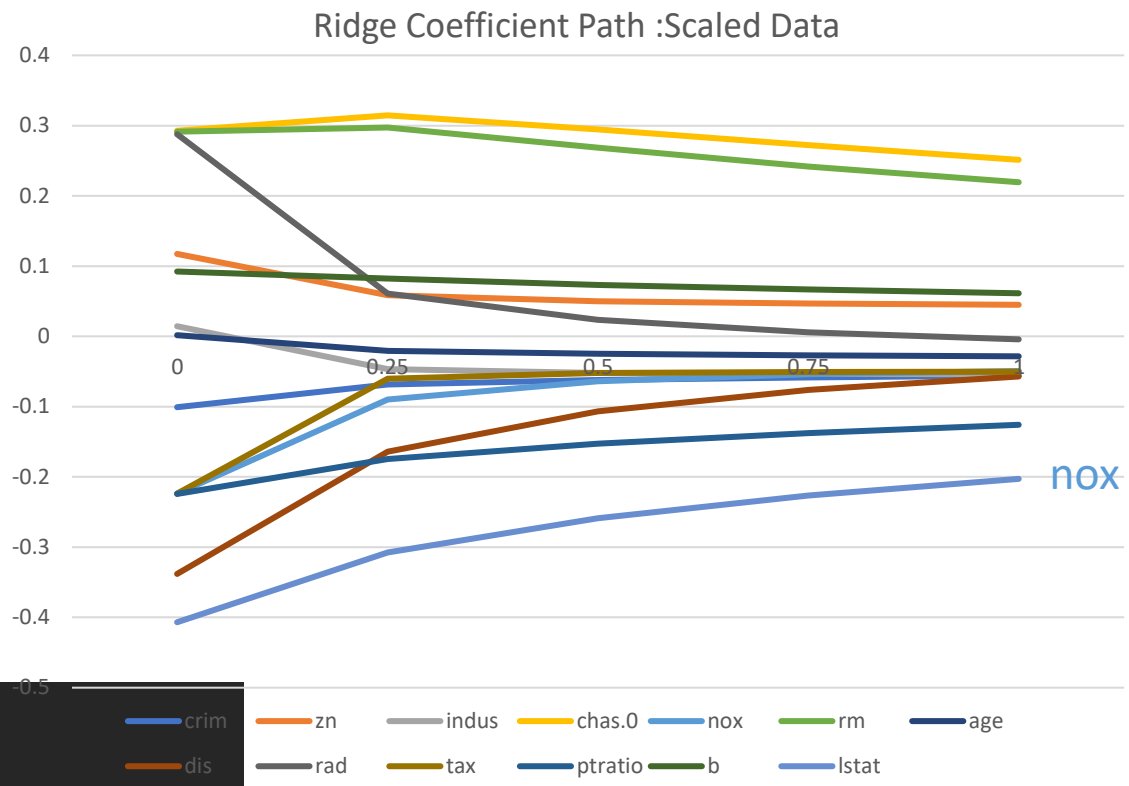
Data มันหน่วยเล็ก -> coefficient มักจะใหญ่ ->
โดน ridge regression กดให้มันเล็ก และจะโดนกดมากกว่าตัวอื่น
แก้ด้วย feature scaling

```
summary(BostonHousing)
      crim                zn             indus       chas        nox               rm             age              dis
 Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   0:471   Min.   :0.3850   Min.   :3.561   Min.   :  2.90   Min.   : 1.130
 1st Qu.: 0.08204   1st Qu.:  0.00   1st Qu.: 5.19   1: 35   1st Qu.:0.4490   1st Qu.:5.886   1st Qu.: 45.02   1st Qu.: 2.100
 Median : 0.25651   Median :  0.00   Median : 9.69           Median :0.5380   Median :6.208   Median : 77.50   Median : 3.207
 Mean   : 3.61352   Mean   : 11.36   Mean   :11.14           Mean   :0.5547   Mean   :6.285   Mean   : 68.57   Mean   : 3.795
 3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10           3rd Qu.:0.6240   3rd Qu.:6.623   3rd Qu.: 94.08   3rd Qu.: 5.188
 Max.   :88.97620   Max.   :100.00   Max.   :27.74           Max.   :0.8710   Max.   :8.780   Max.   :100.00   Max.   :12.127
      rad              tax           ptratio            b               lstat            medv
 Min.   : 1.000   Min.   :187.0   Min.   :12.60   Min.   :  0.32   Min.   : 1.73   Min.   : 5.00
 1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40   1st Qu.:375.38   1st Qu.: 6.95   1st Qu.:17.02
 Median : 5.000   Median :330.0   Median :19.05   Median :391.44   Median :11.36   Median :21.20
 Mean   : 9.549   Mean   :408.2   Mean   :18.46   Mean   :356.67   Mean   :12.65   Mean   :22.53
 3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:396.23   3rd Qu.:16.95   3rd Qu.:25.00
 Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :396.90   Max.   :37.97   Max.   :50.00
```

Linear regression theory (Ridge Regression)

ข้อมูลที่ scaled แล้ว (Standard Scaler)

coef/lambda	0	0.25	0.5	0.75	1
crim	-0.10077	-0.06847	-0.062	-0.05853	-0.05597
zn	0.117484	0.058769	0.049972	0.046714	0.044863
indus	0.014549	-0.04646	-0.0519	-0.05355	-0.05394
chas.0	0.292629	0.314613	0.294609	0.272173	0.251162
nox	-0.2239	-0.08983	-0.06374	-0.05406	-0.04937
rm	0.291319	0.297437	0.268611	0.241951	0.219584
age	0.001822	-0.0206	-0.02467	-0.02695	-0.02846
dis	-0.33814	-0.16409	-0.10686	-0.07641	-0.05708
rad	0.287755	0.061098	0.023416	0.005955	-0.00422
tax	-0.22406	-0.06044	-0.05199	-0.05054	-0.05008
ptratio	-0.22412	-0.17459	-0.15281	-0.1377	-0.12603
b	0.092357	0.082466	0.073351	0.066617	0.061482
lstat	-0.40713	-0.30779	-0.25876	-0.22645	-0.20287




```
library(caret)
scale_param <- preProcess(BostonHousing , method = c("scale", "center"))
BostonHousing_transform <- predict(scale_param, BostonHousing)
summary(BostonHousing_transform)
      crim                zn               indus          chas          nox                rm               age               dis
 Min.   :-0.419367   Min.   :-0.48724   Min.   :-1.5563   0:471   Min.   :-1.4644   Min.   :-3.8764   Min.   :-2.3331   Min.   :-1.2658
 1st Qu.:-0.410563   1st Qu.:-0.48724   1st Qu.:-0.8668   1: 35   1st Qu.:-0.9121   1st Qu.:-0.5681   1st Qu.:-0.8366   1st Qu.:-0.8049
 Median :-0.390280   Median :-0.48724   Median :-0.2109           Median :-0.1441   Median :-0.1084   Median : 0.3171   Median :-0.2790
 Mean   : 0.000000   Mean   : 0.00000   Mean   : 0.0000           Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
 3rd Qu.: 0.007389   3rd Qu.: 0.04872   3rd Qu.: 1.0150           3rd Qu.: 0.5981   3rd Qu.: 0.4823   3rd Qu.: 0.9059   3rd Qu.: 0.6617
 Max.   : 9.924110   Max.   : 3.80047   Max.   : 2.4202           Max.   : 2.7296   Max.   : 3.5515   Max.   : 1.1164   Max.   : 3.9566
      rad               tax             ptratio              b               lstat              medv
 Min.   :-0.9819   Min.   :-1.3127   Min.   :-2.7047   Min.   :-3.9033   Min.   :-1.5296   Min.   :-1.9063
 1st Qu.:-0.6373   1st Qu.:-0.7668   1st Qu.:-0.4876   1st Qu.: 0.2049   1st Qu.:-0.7986   1st Qu.:-0.5989
 Median :-0.5225   Median :-0.4642   Median : 0.2746   Median : 0.3808   Median :-0.1811   Median :-0.1449
 Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
 3rd Qu.: 1.6596   3rd Qu.: 1.5294   3rd Qu.: 0.8058   3rd Qu.: 0.4332   3rd Qu.: 0.6024   3rd Qu.: 0.2683
 Max.   : 1.6596   Max.   : 1.7964   Max.   : 1.6372   Max.   : 0.4406   Max.   : 3.5453   Max.   : 2.9865
> apply(BostonHousing_transform,2,sd)
    crim       zn    indus     chas      nox       rm      age      dis      rad      tax  ptratio        b    lstat     medv
1.000000 1.000000 1.000000 0.253994 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
>
```

Lasso regression (L2)

cost function ของ regression model เดิมคือ **sum of squared error**

Linear Regression Cost = $\sum(\hat{y}_i - y_i)$ ^2

Lasso regression ใส่ ผลรวมของ **Absolute coefficient** เข้าไปใน cost function

Lasso Regression Cost = $\sum(\hat{y}_i - y_i)$ ^2 + $\lambda \sum$ |bj|

- **ถ้า feature ที่เพิ่มเข้ามาเยอะมา**ก -> ก้อนหลังเยอะขึ้น -> **total cost เราเยอะขึ้น**
- เนื่องจากเรา**อยาก minimize cost** -> lasso regression ทำให้ค่าของ coefficient หลายตัวเป็น 0 -> **จำนวน feature น้อยลง**

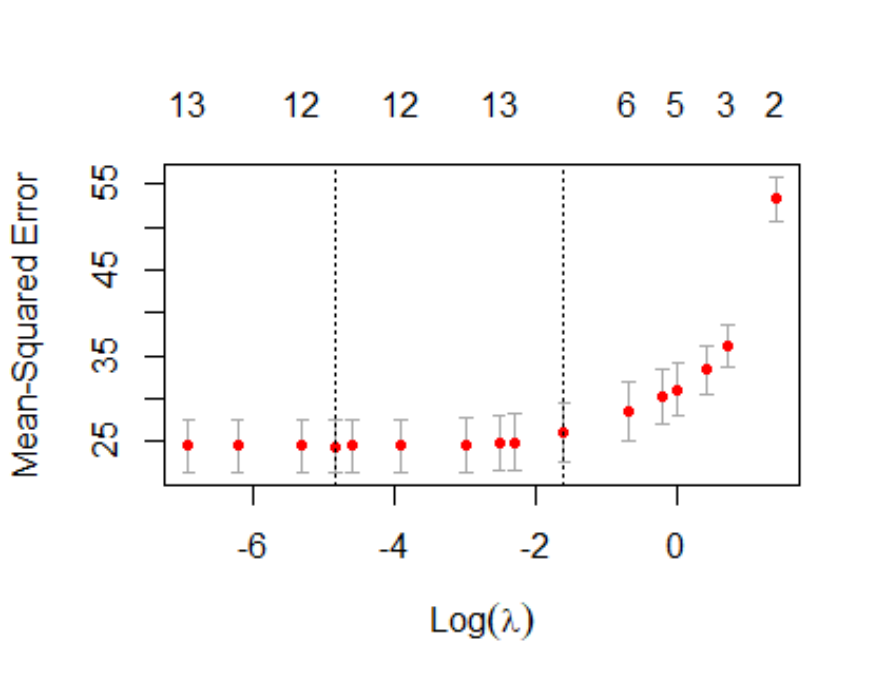
Linear regression theory (Lasso Regression)

```
library(dplyr)
library(glmnet)
library(caret)
y_train <- training_scaled$medv
x_train <- training_scaled[,-which(names(training_scaled) == "medv")] %>% data.matrix()

y_test <- testing_scaled$medv
x_test <- testing_scaled[,-which(names(testing_scaled) == "medv")] %>% data.matrix()

lasso_model <- glmnet(x_train , y_train , alpha = 1, lambda = 0.02)
y_pred <-  predict(lasso_model , newx = x_test)
RMSE(y_pred , y_test)
[1] 4.427482
coef(lasso_model)
14 x 1 sparse Matrix of class "dgCMatrix"
                    s0
(Intercept) 20.1627172
crim        -0.8888490
zn           1.1603312
indus        .
chas         2.3857530
nox         -2.1037655
rm           2.4816141
age          0.2708545
dis         -3.1725466
rad          2.7490942
tax         -1.8456581
ptratio     -2.1969865
b            0.9065252
lstat       -4.2323624
```

glmnet data จะต้องอยู่ในรูป matrix !!!

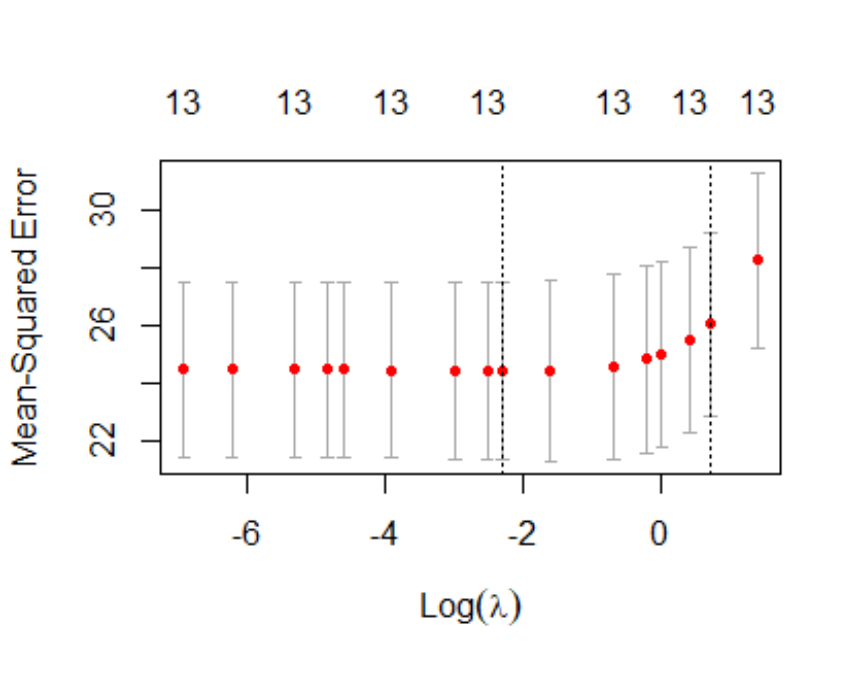
Linear regression theory (Lasso Regression)

```
set.seed(1689)
lambda_set = c(0.001,0.002,0.005,0.008,0.01,0.02,0.05,0.08,0.1,0.2,0.5,0.8,1,1.5,2,4)
cv_lasso <- cv.glmnet(x_train, y_train ,nfolds = 5, alpha = 1, lambda = lambda_set)
plot(cv_lasso)
best_lambda <- cv_lasso$lambda.min
best_lambda
[1] 0.008
```



Linear regression theory (Ridge Regression R code)

```
set.seed(1689)
lambda_set = c(0.001,0.002,0.005,0.008,0.01,0.02,0.05,0.08,0.1,0.2,0.5,0.8,1,1.5,2,4)
cv_ridge <- cv.glmnet(x_train, y_train ,nfolds = 5, alpha = 0, lambda = lambda_set)
plot(cv_ridge)
best_lambda <- cv_ridge$lambda.min
best_lambda
[1] 0.1
```



Caret Version

```
library(caret)
library(glmnet)
lasso_model <- train(
                medv ~ .,
                        data = BostonHousing,
                        method = "glmnet",
                        tuneGrid = expand.grid(
                                alpha = 1,
                                lambda = 0.2
                                                )
                    )
lasso_model
```

glmnet model
alpha (ridge ให้ set = 0, lasso ให้ set = 1)
Lambda (normal linear regression ให้ set = 0 , ถ้าจะ penalized parameter ให้ set >0 ยิ่งเยอะยิ่ง penalized แรง**)**

Caret Version

```
library(caret)
library(glmnet)
lasso_model <- train(
                medv ~ .,
                        data = BostonHousing,
                        method = "glmnet",
                        tuneGrid = expand.grid(
                                alpha = 1,
                                lambda = 0.2
                                                )
                    )
lasso_model
```

หากต้องการ tune hyperparameter alpha หรือ lambda ให้ใส่ value เข้าไปเป็น vector

เช่น

alpha = c(0,0.1,0.2,0.5,0.8,1)

lambda = c(0,0.01,0.02,0.05,0.1,0.5,1,2)

glmnet model

alpha (ridge ให้ set = 0, lasso ให้ set = 1)

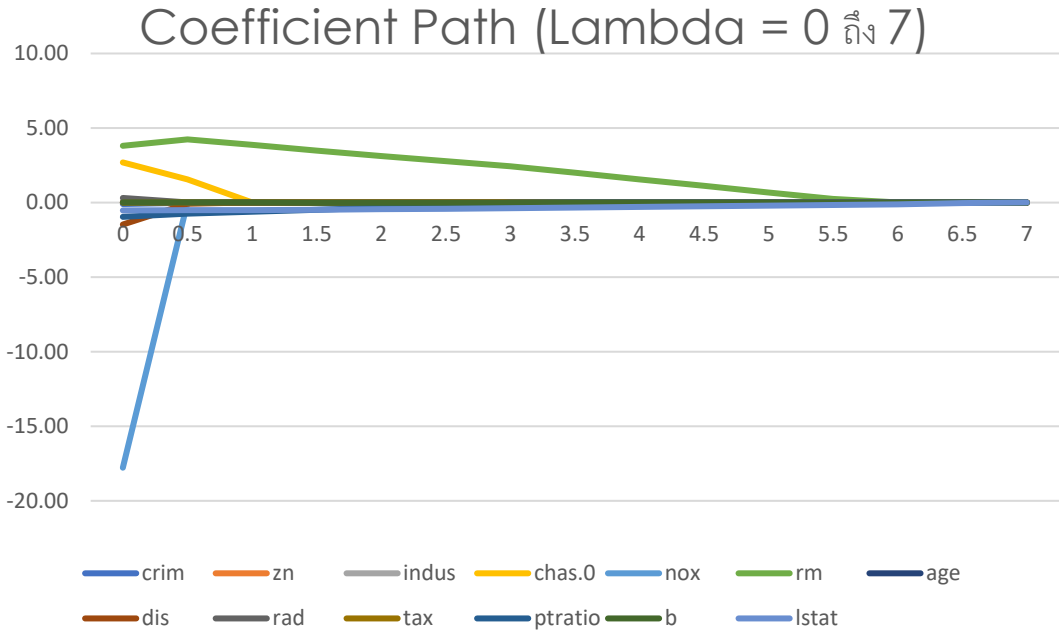
Lambda (normal linear regression ให้ set = 0 , ถ้าจะ penalized parameter ให้ set >0 ยิ่งเยอะยิ่ง penalized แรง)

Linear regression theory (Lasso Regression)

Lambda

Ridge Regression Cost $= \sum(\hat{y}_i - y_i)^2 + \lambda \sum bj^2$

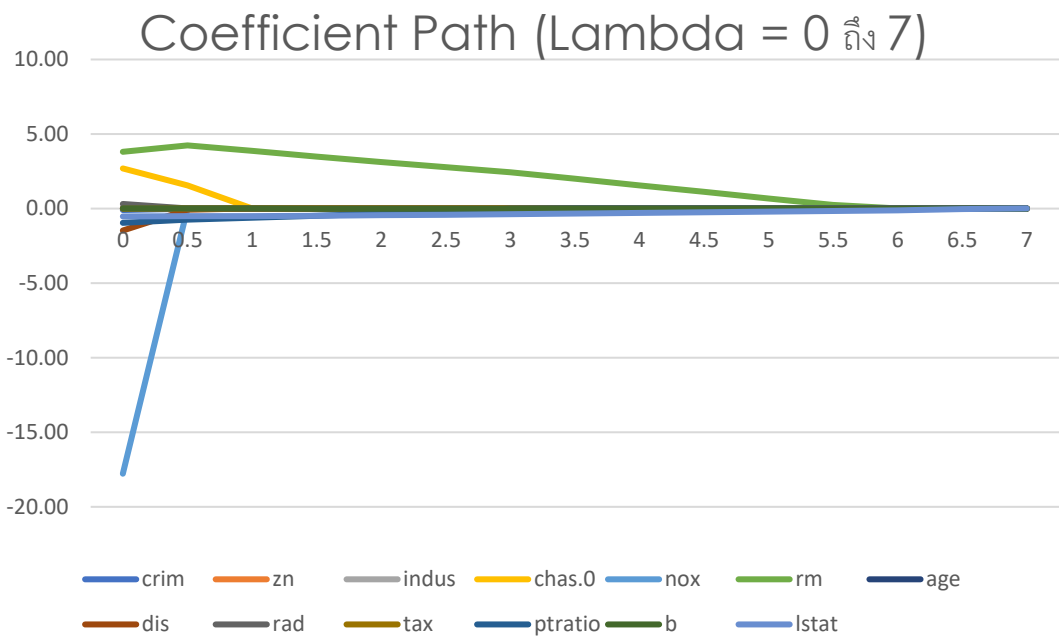
coef/lambda	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7
crim	-0.11	-0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
zn	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
indus	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
chas.0	2.69	1.56	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
nox	-17.77	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
rm	3.81	4.24	3.86	3.47	3.13	2.78	2.43	2.01	1.56	1.12	0.68	0.24	0.00	0.00	0.00
age	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
dis	-1.48	-0.08	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
rad	0.30	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
tax	-0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
ptratio	-0.95	-0.74	-0.62	-0.48	-0.32	-0.17	-0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
b	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
lstat	-0.52	-0.51	-0.50	-0.48	-0.44	-0.41	-0.38	-0.34	-0.29	-0.25	-0.21	-0.16	-0.11	-0.04	0.00




```
summary(BostonHousing)
      crim                zn             indus       chas        nox               rm             age              dis
 Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   0:471   Min.   :0.3850   Min.   :3.561   Min.   :  2.90   Min.   : 1.130
 1st Qu.: 0.08204   1st Qu.:  0.00   1st Qu.: 5.19   1: 35   1st Qu.:0.4490   1st Qu.:5.886   1st Qu.: 45.02   1st Qu.: 2.100
 Median : 0.25651   Median :  0.00   Median : 9.69           Median :0.5380   Median :6.208   Median : 77.50   Median : 3.207
 Mean   : 3.61352   Mean   : 11.36   Mean   :11.14           Mean   :0.5547   Mean   :6.285   Mean   : 68.57   Mean   : 3.795
 3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10           3rd Qu.:0.6240   3rd Qu.:6.623   3rd Qu.: 94.08   3rd Qu.: 5.188
 Max.   :88.97620   Max.   :100.00   Max.   :27.74           Max.   :0.8710   Max.   :8.780   Max.   :100.00   Max.   :12.127
      rad              tax           ptratio            b               lstat            medv
 Min.   : 1.000   Min.   :187.0   Min.   :12.60   Min.   :  0.32   Min.   : 1.73   Min.   : 5.00
 1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40   1st Qu.:375.38   1st Qu.: 6.95   1st Qu.:17.02
 Median : 5.000   Median :330.0   Median :19.05   Median :391.44   Median :11.36   Median :21.20
 Mean   : 9.549   Mean   :408.2   Mean   :18.46   Mean   :356.67   Mean   :12.65   Mean   :22.53
 3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:396.23   3rd Qu.:16.95   3rd Qu.:25.00
 Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :396.90   Max.   :37.97   Max.   :50.00
```

Linear regression theory (Lasso Regression)

coef/lambda	0	0.5	1	1.5	2	2.5	3	3.5	4	4.5	5	5.5	6	6.5	7
crim	-0.11	-0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
zn	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
indus	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
chas.0	2.69	1.56	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
nox	-17.77	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
rm	3.81	4.24	3.86	3.47	3.13	2.78	2.43	2.01	1.56	1.12	0.68	0.24	0.00	0.00	0.00
age	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
dis	-1.48	-0.08	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
rad	0.30	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
tax	-0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
ptratio	-0.95	-0.74	-0.62	-0.48	-0.32	-0.17	-0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
b	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
lstat	-0.52	-0.51	-0.50	-0.48	-0.44	-0.41	-0.38	-0.34	-0.29	-0.25	-0.21	-0.16	-0.11	-0.04	0.00



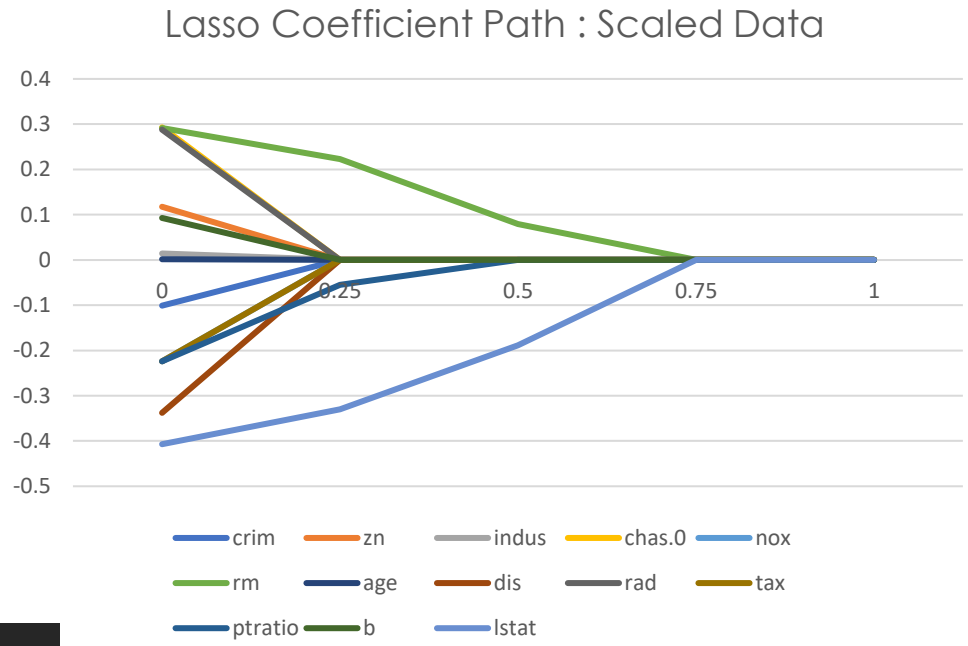
Data มันหน่วยเล็ก -> coefficient มักจะใหญ่ ->
โดน ridge regression กดให้มันเล็ก และจะโดนกดมากกว่าตัวอื่น
แก้ด้วย feature scaling

```
summary(BostonHousing)
      crim                zn             indus       chas        nox               rm             age              dis
 Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   0:471   Min.   :0.3850   Min.   :3.561   Min.   :  2.90   Min.   : 1.130
 1st Qu.: 0.08204   1st Qu.:  0.00   1st Qu.: 5.19   1: 35   1st Qu.:0.4490   1st Qu.:5.886   1st Qu.: 45.02   1st Qu.: 2.100
 Median : 0.25651   Median :  0.00   Median : 9.69           Median :0.5380   Median :6.208   Median : 77.50   Median : 3.207
 Mean   : 3.61352   Mean   : 11.36   Mean   :11.14           Mean   :0.5547   Mean   :6.285   Mean   : 68.57   Mean   : 3.795
 3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10           3rd Qu.:0.6240   3rd Qu.:6.623   3rd Qu.: 94.08   3rd Qu.: 5.188
 Max.   :88.97620   Max.   :100.00   Max.   :27.74           Max.   :0.8710   Max.   :8.780   Max.   :100.00   Max.   :12.127
      rad              tax           ptratio            b               lstat            medv
 Min.   : 1.000   Min.   :187.0   Min.   :12.60   Min.   :  0.32   Min.   : 1.73   Min.   : 5.00
 1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40   1st Qu.:375.38   1st Qu.: 6.95   1st Qu.:17.02
 Median : 5.000   Median :330.0   Median :19.05   Median :391.44   Median :11.36   Median :21.20
 Mean   : 9.549   Mean   :408.2   Mean   :18.46   Mean   :356.67   Mean   :12.65   Mean   :22.53
 3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:396.23   3rd Qu.:16.95   3rd Qu.:25.00
 Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :396.90   Max.   :37.97   Max.   :50.00
```

Linear regression theory (Lasso Regression)

ข้อมูลที่ scaled แล้ว (Standard Scaler)

coef/lambda	0	0.25	0.5	0.75	1
crim	-0.10077	0	0	0	0
zn	0.117484	0	0	0	0
indus	0.014549	0	0	0	0
chas.0	0.292629	0	0	0	0
nox	-0.2239	0	0	0	0
rm	0.291319	0.223093	0.079133	0	0
age	0.001822	0	0	0	0
dis	-0.33814	0	0	0	0
rad	0.287755	0	0	0	0
tax	-0.22406	0	0	0	0
ptratio	-0.22412	-0.05482	0	0	0
b	0.092357	0	0	0	0
lstat	-0.40713	-0.32998	-0.1886	0	0




```
library(caret)
scale_param <- preProcess(BostonHousing , method = c("scale", "center"))
BostonHousing_transform <- predict(scale_param, BostonHousing)
summary(BostonHousing_transform)
      crim                zn               indus         chas         nox                rm               age               dis
 Min.   :-0.419367   Min.   :-0.48724   Min.   :-1.5563   0:471   Min.   :-1.4644   Min.   :-3.8764   Min.   :-2.3331   Min.   :-1.2658
 1st Qu.:-0.410563   1st Qu.:-0.48724   1st Qu.:-0.8668   1: 35   1st Qu.:-0.9121   1st Qu.:-0.5681   1st Qu.:-0.8366   1st Qu.:-0.8049
 Median :-0.390280   Median :-0.48724   Median :-0.2109           Median :-0.1441   Median :-0.1084   Median : 0.3171   Median :-0.2790
 Mean   : 0.000000   Mean   : 0.00000   Mean   : 0.0000           Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
 3rd Qu.: 0.007389   3rd Qu.: 0.04872   3rd Qu.: 1.0150           3rd Qu.: 0.5981   3rd Qu.: 0.4823   3rd Qu.: 0.9059   3rd Qu.: 0.6617
 Max.   : 9.924110   Max.   : 3.80047   Max.   : 2.4202           Max.   : 2.7296   Max.   : 3.5515   Max.   : 1.1164   Max.   : 3.9566
      rad               tax             ptratio              b                lstat              medv
 Min.   :-0.9819   Min.   :-1.3127   Min.   :-2.7047   Min.   :-3.9033   Min.   :-1.5296   Min.   :-1.9063
 1st Qu.:-0.6373   1st Qu.:-0.7668   1st Qu.:-0.4876   1st Qu.: 0.2049   1st Qu.:-0.7986   1st Qu.:-0.5989
 Median :-0.5225   Median :-0.4642   Median : 0.2746   Median : 0.3808   Median :-0.1811   Median :-0.1449
 Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000
 3rd Qu.: 1.6596   3rd Qu.: 1.5294   3rd Qu.: 0.8058   3rd Qu.: 0.4332   3rd Qu.: 0.6024   3rd Qu.: 0.2683
 Max.   : 1.6596   Max.   : 1.7964   Max.   : 1.6372   Max.   : 0.4406   Max.   : 3.5453   Max.   : 2.9865
> apply(BostonHousing_transform,2,sd)
    crim       zn    indus     chas      nox       rm      age      dis      rad      tax  ptratio        b    lstat     medv
1.000000 1.000000 1.000000 0.253994 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000 1.000000
>
```

Linear regression theory (Lasso Regression)

```
coef(glmnet(x_train , y_train  ,alpha = 1,lambda = 0.008 ))
14 x 1 sparse Matrix of class "dgCMatrix"
                    s0
(Intercept) 20.1584477
crim        -0.9264323
zn           1.2078000
indus        .
chas         2.3897295
nox         -2.1668578
rm           2.4578186
age          0.3239085
dis         -3.2381321
rad          2.9245208
tax         -1.9865859
ptratio     -2.2156977
b            0.9155138
lstat       -4.2477386
coef(glmnet(x_train , y_train  ,alpha = 1,lambda = 0.5 ))
14 x 1 sparse Matrix of class "dgCMatrix"
                    s0
(Intercept) 21.1766419
crim         .
zn           .
indus        .
chas         1.4414342
nox          .
rm           2.7937947
age          .
dis         -0.3712142
rad          .
tax          .
ptratio     -1.6676076
b            0.5601777
lstat       -4.1427048
```

เพิ่ม lambda -> beta เป็น 0 เยอะขึ้น

LASSO FEATURE SELECTION FOR MULTIPLE LINEAR REGRESISON

เลือก feature ที่เราต้องการออกมาเพื่อ

1. ลดจำนวน feature เพื่อให้ อธิบาย model ทางธุรกิจได้ดีขึ้น
2. ลดการ overfit model
3. ลดปัญหา multicollinearity

ทำให้ Features ต่าง ๆมีหน่วยระดับเดียวกัน -> scaler

```
library(caret)
num_train <- createDataPartition(y= BostonHousing$medv , p =0.70 , list = FALSE)
train_data <- BostonHousing[num_train,]
test_data <- BostonHousing[-num_train,]
```

Popular scaler จะมีอยู่ 2 ตัว

- 1. standard scaler : ทำให้ข้อมูลแต่ละ Features มีค่า mean = 0 , variance = 1

สูตร

$$x_{scaled} = \frac{x - mean}{sd}$$

```
library(caret)
num_train <- createDataPartition(y= BostonHousing$medv , p =0.70 , list = FALSE)
train_data <- BostonHousing[num_train,]
test_data <- BostonHousing[-num_train,]
train_feature <- train_data[, -which(names(train_data) == "medv")]
scaling_model <- preProcess(train_feature, method = c("center","scale"))
train_scaled <- predict(scaling_model , newdata =train_data)
test_scaled <- predict(scaling_model , newdata =test_data)
```

- 2. min max scaler : ทำให้ข้อมูลแต่ละ Features มีค่าอยู่ระหว่าง {0 – 1}

สูตร

$$X_{new} = \frac{X_i - \min(X)}{\max(x) - \min(X)}$$

```
library(caret)
num_train <- createDataPartition(y= BostonHousing$medv , p =0.70 , list = FALSE)
train_data <- BostonHousing[num_train,]
test_data <- BostonHousing[-num_train,]
train_feature <- train_data[, -which(names(train_data) == "medv")]
scaling_model <- preProcess(train_feature, method = c("range"))
train_scaled <- predict(scaling_model , newdata =train_data)
test_scaled <- predict(scaling_model , newdata =test_data)
```

Elasticnet Regression คือการผสมระหว่าง **Ridge regression** + **Lasso Regression**

Elasticnet regression Cost :

$$\frac{\sum_{i=1}^{n}(y_i - x_i^J \hat{\beta})^2}{2n} + \lambda \left(\frac{1-\alpha}{2} \sum_{j=1}^{m} \hat{\beta}_j^2 + \alpha \sum_{j=1}^{m} |\hat{\beta}_j| \right)$$

- ใช้ค่า **lambda** เป็นตัว weight penalized parameter หากมีค่าเยอะ จะ penalized เยอะ (ลดค่า beta เยอะ) หากน้อย จะ penalized น้อย (ลดค่า beta น้อย)
  - normal regression (lambda = 0) และ regularized regression (lambda > 0)
- ใช้ค่า **alpha** เป็นตัว weight ระหว่าง ridge regression and lasso regression
  - ridge regression (alpha = 0) , lasso regression (alpha = 1) , elasticnet regression (alpha อยู่ระหว่าง 0 ถึง 1)
- **ถ้า feature ที่เพิ่มเข้ามาเยอะมาก** -> ก้อนหลังเยอะขึ้น -> **total cost เราเยอะขึ้น**
- เนื่องจากเรา**อยาก minimize cost** -> elasticnet regression ทำให้ค่าของ coefficient หลายตัวเป็น หรือ เข้าใกล้ 0 มากขึ้น

Linear regression theory (Elasticnet Regression)

```
library(dplyr)
library(glmnet)
library(caret)
y_train <- training_scaled$medv
x_train <- training_scaled[,-which(names(training_scaled) == "medv")] %>% data.matrix()
y_test <- testing_scaled$medv
x_test <- testing_scaled[,-which(names(testing_scaled) == "medv")] %>% data.matrix()
set.seed(1689)
elasticnet_model <- glmnet(x_train , y_train ,alpha = 0.5, lambda = 0.02)
y_pred <-  predict(elasticnet_model , newx = x_test)
RMSE(y_pred , y_test)
[1] 4.443024
coef(elasticnet_model)
14 x 1 sparse Matrix of class "dgCMatrix"
                    s0
(Intercept) 20.1524242
crim        -0.9149188
zn           1.1924018
indus        .
chas         2.3953395
nox         -2.1451212
rm           2.4681084
age          0.3095748
dis         -3.2105122
rad          2.8600353
tax         -1.9341133
ptratio     -2.2086518
b            0.9140441
lstat       -4.2374414
```

Linear regression theory (Elasticnet Regression)

Caret Version

```
library(caret)
library(glmnet)
elas_model <- train(
                medv ~ .,
                        data = BostonHousing,
                        method = "glmnet",
                        tuneGrid = expand.grid(
                                alpha = 0.5,
                                lambda = 0.2
                                                )
                    )
elas_model
```

glmnet model
alpha (ridge ให้ set = 0, lasso ให้ set = 1 , elasticnet ให้ set ระหว่าง 0 ถึง 1 เช่น 0.5)
Lambda (normal linear regression ให้ set = 0 , ถ้าจะ penalized parameter ให้ set >0 ยิ่งเยอะยิ่ง penalized แรง**)**

Caret Version

```
library(caret)
library(glmnet)
elas_model <- train(
                medv ~ .,
                        data = BostonHousing,
                        method = "glmnet",
                        tuneGrid = expand.grid(
                                alpha = 0.5,
                                lambda = 0.2
                                                )
                    )
elas_model
```

หากต้องการ tune hyperparameter alpha หรือ lambda ให้ใส่ value เข้าไปเป็น vector

เช่น

alpha = c(0,0.1,0.2,0.5,0.8,1)

lambda = c(0,0.01,0.02,0.05,0.1,0.5,1,2)

glmnet model

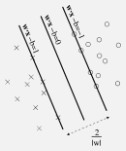
alpha (ridge ให้ set = 0, lasso ให้ set = 1 , elasticnet ให้ set ระหว่าง 0 ถึง 1 เช่น 0.5)

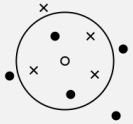
Lambda (normal linear regression ให้ set = 0 , ถ้าจะ penalized parameter ให้ set >0 ยิ่งเยอะยิ่ง penalized แรง)

R Session Multiple Linear regression

Various Type of Regression

Type of regression modeling

- **1. Linear regression**
- **2. Tree regression**
- **3. Support vector regression** 

- **4. Distance regression** 

Type of regression modeling

- **1. Linear regression**
  - **1.1 Simple linear regression** ✓
  - **1.2 Multiple linear regression** ✓
  - **1.3 Polynomial regression** ✓

- **2. Tree regression**
  - **2.1 Decision Tree regression**
  - **2.2 Random Forest regression**
  - **2.3 Gradient Boosting regression**

Type of regression modeling

- **3. Support vector regression**
    - **3.1 Support vector regression : linear kernel**
    - **3.2 Support vector regression : kernel trick**
- **4. Distance regression**
    - **4.1 K-Nearest Neighbor regression**
- **5. Ensemble regression**

Tree Regression

Tree Regression (Fundamental)

Linear Regression

ตีสมการเส้นตรง สร้างความสัมพันธ์
ระหว่าง Feature กับ Target

VS

Decision Tree Regression

ไม่มีสมการ แต่ใช้หลักการ **If-Else ในการ split ข้อมูล**
เพื่อ minimize Regression Cost
โดยการ split ข้อมูลแต่ละครั้ง จะใช้

1. Cut off ของ data point
2. Feature
ในการ split

Tree Regression (Decision Tree)

Linear Regression : Y = b0 + b1*X1 +b2*X2
Linear Regression : Weight = b0 + b1*Gender_dummy + b2*height

สมมุติ เราต้องการทำ Model Predict Weight
Target (Y) : Weight
Feature (X) : Gender , height

weight	gender	height
55	F	165
45	F	159
41	F	151
60	M	165
70	M	171
75	M	180

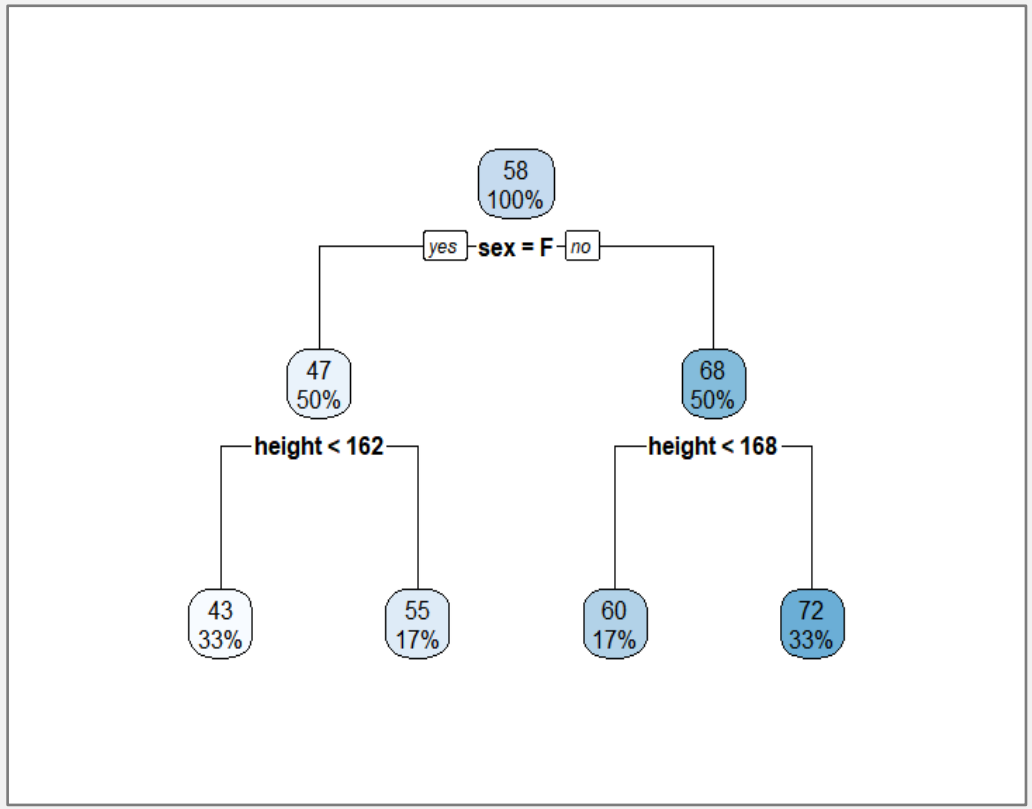
```
weight <- c(55 , 45,41,60,70,75)
sex <- c('F','F','F','M','M','M')
height <- c(165,159,151,165,171,180)
weight_data <- data.frame(weight, sex, height)
str(weight_data)
'data.frame':	6 obs. of  3 variables:
 $ weight: num  55 45 41 60 70 75
 $ sex   : Factor w/ 2 levels "F","M": 1 1 1 2 2 2
 $ height: num  165 159 151 165 171 180
```

Tree Regression (Decision Tree Example)

สมมุติ เราต้องการทำ Model Predict Weight

Target (Y) : Weight

Feature (X) : Gender , height



weight	gender	height
55	F	165
45	F	159
41	F	151
60	M	165
70	M	171
75	M	180

หากเรามีข้อมูล gender = F , height = 152

แล้วต้องการ prediction weight เราสามารถใส่เลขเข้าไปใน model ได้

Sex = F (yes) และ height < 162

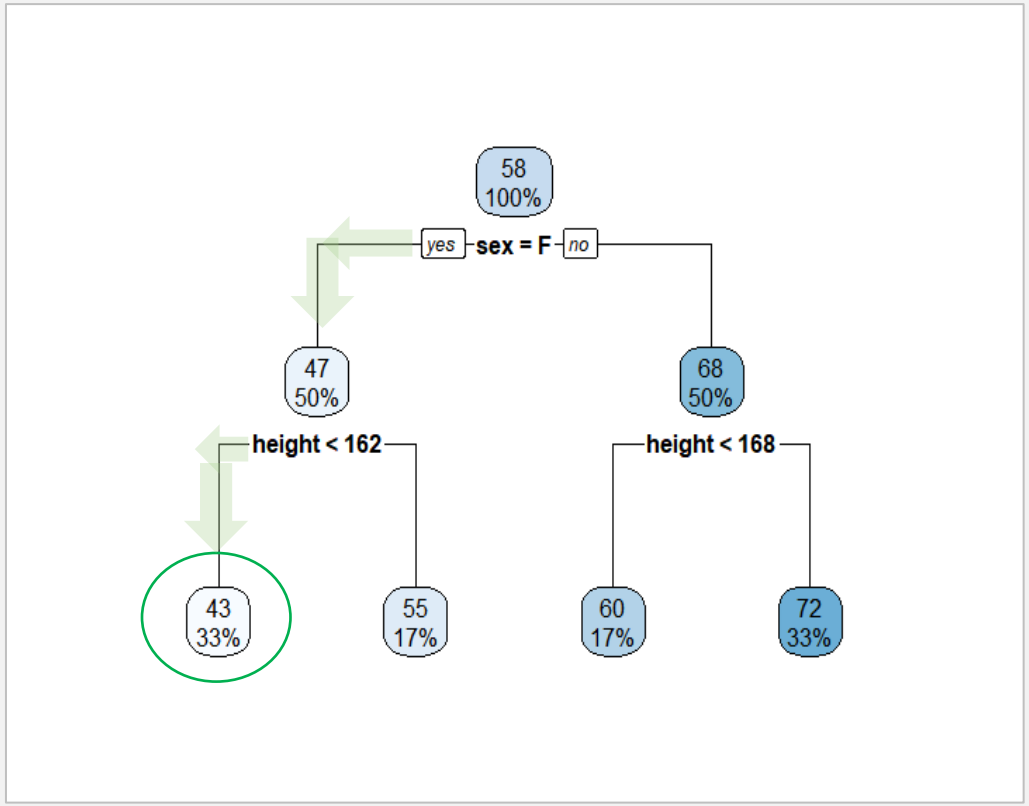
เราจะได้ค่า prediction weight = 43

Tree Regression (Decision Tree Example)

สมมุติ เราต้องการทำ Model Predict Weight

Target (Y) : Weight

Feature (X) : Gender , height



หากเรามีข้อมูล gender = F , height = 152

แล้วต้องการ prediction weight เราสามารถใส่เลขเข้าไปใน model ได้

Sex = F (yes) และ height < 162

เราจะได้ค่า prediction weight = 43

Tree Regression (Decision Tree Example)

Linear Regression

ตีสมการเส้นตรง สร้างความสัมพันธ์
ระหว่าง Feature กับ Target

VS

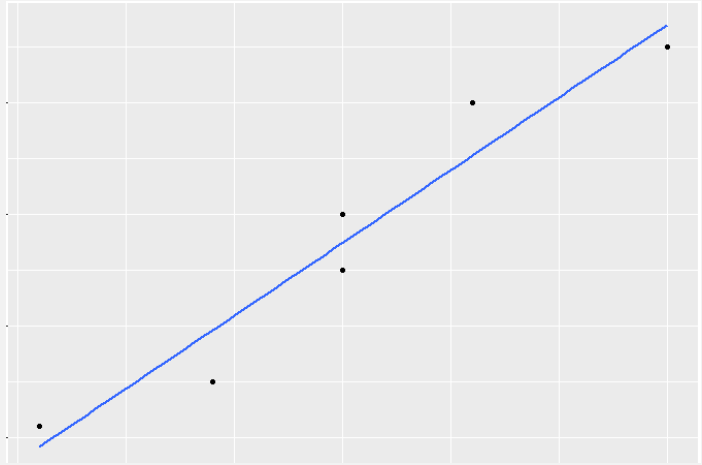
Decision Tree Regression

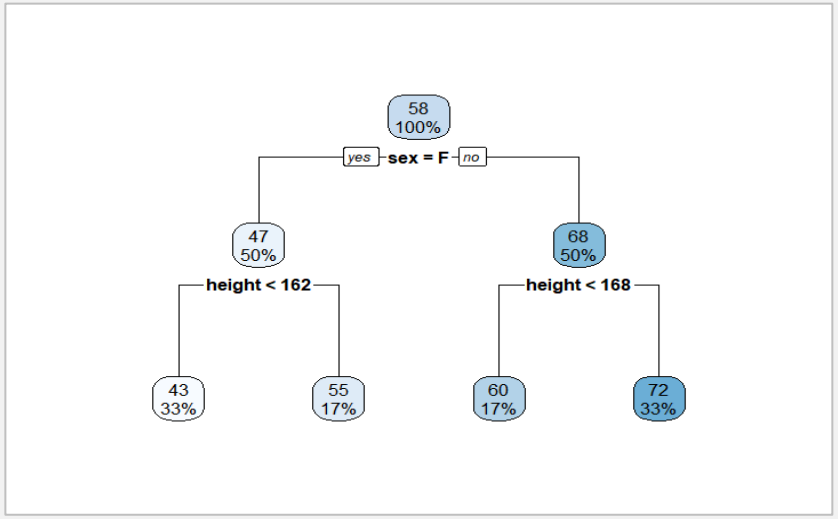
ไม่มีสมการ แต่ใช้หลักการ If-Else ในการ split ข้อมูล
เพื่อ minimize Regression Cost
โดยการ split ข้อมูลแต่ละครั้ง จะใช้

1. Cut off จุดกึ่งกลางระหว่าง data point
2. Feature

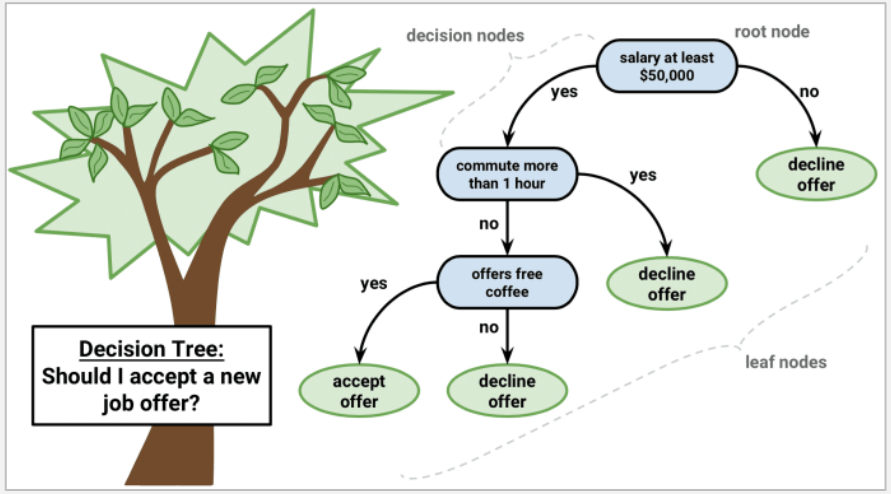
ในการ split
โดยจะ split ไปเรื่อยๆ จนกว่าจะถึง criteria ที่เรากำหนด

Linear Regression : Y = b0 + b1*X1 +b2*X2
Linear Regression : Weight = b0 + b1*Gender_dummy + b2*height

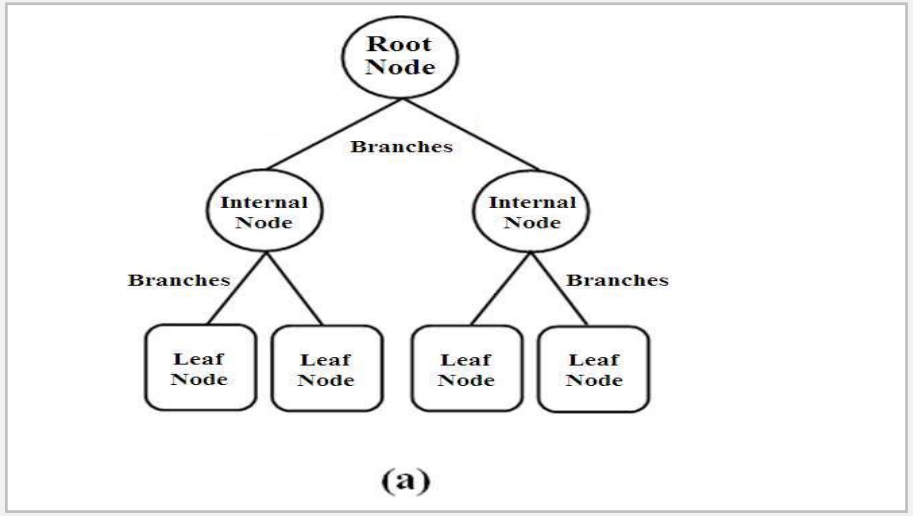




Tree Regression (Term Definition)



https://mc.ai/mathematics-behind-decision-tree/

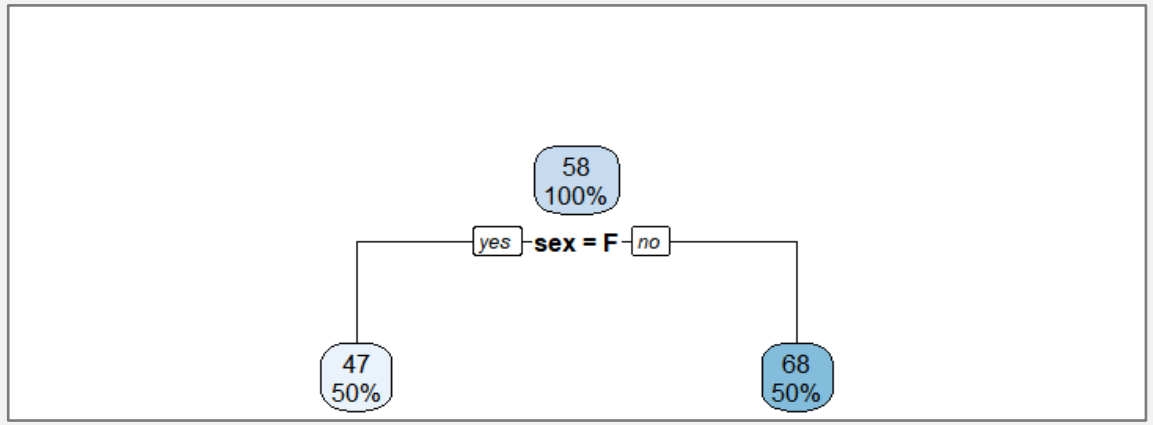


https://www.researchgate.net/figure/a-describes-the-components-of-a-decision-tree-the-Nodes-represent-the-possible_fig2_303773171

1. **Leaf Node** : Node สุดท้าย ที่สิ้นสุดการ split แล้ว
2. **Root Node** : Node แรกก่อนที่จะเริ่ม split ทุกตัว
3. **Split** : การ แบ่งข้อมูลจาก 1 Node เพื่อสร้าง 2 Node หรือมากกว่าต่อไป
4. **Internal Node** : Node ที่อยู่ข้างในระหว่าง Root Node และ Leaf Node
5. **Decision Node** : คือ Node ที่มีการ make decision สำหรับ split (Root Node + Internal Node)
6. **Tree Depth** : ความลึกการ Tree ที่เราสร้าง ยิ่ง split ต่อๆ กันไปเรื่อยๆเยอะ Tree Depth ยิ่งลึก
7. **Pruning** : การลดจำนวน Node (ตรงข้ามกับ split ที่จะสร้าง Node เพิ่ม)
8. **Observation In Node** : ในแต่ละ Node จะบอกจำนวน Data Point ที่อยู่ใน Node นั้นด้วย

Tree Regression (Term Definition) การ split

weight	gender	height
41	F	151
45	F	159
55	F	165
60	M	165
70	M	171
75	M	180

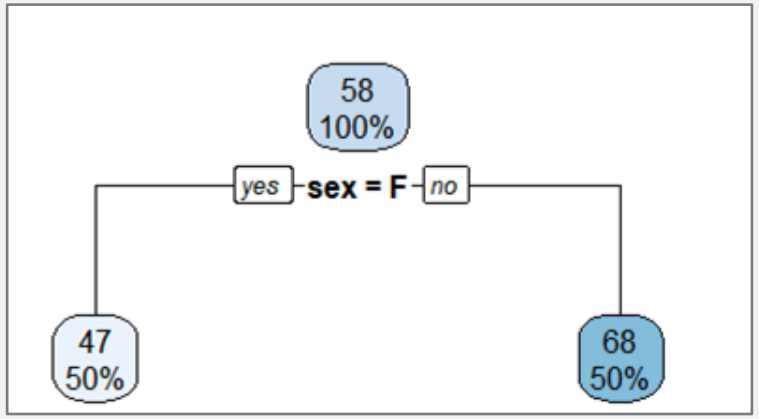


เราสามารถ SPLIT on

1. **Gender (F vs M)**
2. **Height**
   1. มากกว่าหรือเท่ากับ 155 vs น้อยกว่า 155 (cut off = [151+159]/2 =155)
   2. มากกว่าหรือเท่ากับ 162 vs น้อยกว่า 162 (cut off = [159+165]/2 =162)
   3. มากกว่าหรือเท่ากับ 165 หรือ น้อยกว่า 165 (cut off = [165+165]/2 =165)
   4. มากกว่าหรือเท่ากับ 168 หรือ น้อยกว่า 168 (cut off = [165+171]/2 =168)
   5. มากกว่าหรือเท่ากับ 175.5 หรือ น้อยกว่า 175.5 (cut off = [171+180]/2 =175.5)

Tree Regression (Term Definition) การหา leaf value

weight	gender	height
41	F	151
45	F	159
55	F	165
60	M	165
70	M	171
75	M	180



เราสามารถ SPLIT on

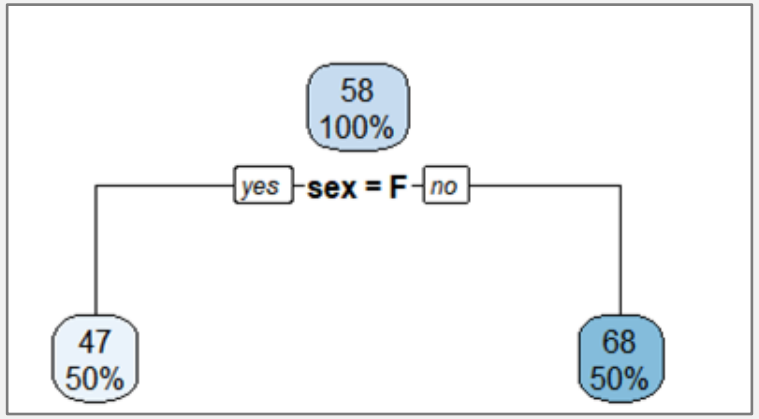
1. **Gender (F vs M)**

   Prediction (ขาแรก sex = F) = mean(41 ,45,55) = 47

   Prediction (ขาแรก sex = M) = mean(60 ,70,75) = 68.333

Tree Regression (Term Definition) การหา cost

weight	gender	height
41	F	151
45	F	159
55	F	165
60	M	165
70	M	171
75	M	180



เราสามารถ SPLIT on

Gender (F vs M)
Prediction (ขาแรก sex = F) = mean(41 ,45,55) = 47
Prediction (ขาแรก sex = M) = mean(60 ,70,75) = 68.333

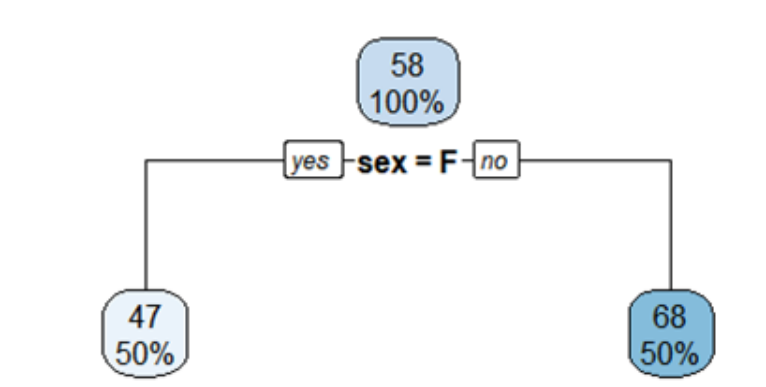
Cost in decision tree : R_squared
หาก Split ต่อไปแล้ว R_squared ที่เพิ่มน้อยกว่าจำนวนที่กำหนด จะเลิก split

วิธีคิด cost : R^2

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$$

Tree Regression (Term Definition) การหา cost

weight	gender	(gender Prediction)	sum of squared error	avg	sum of squared total	rsquared
55	F	47	64.00	57.67	7.11	**0.756**
45	F	47	4.00	57.67	160.44	
41	F	47	36.00	57.67	277.78	
60	M	68.3	69.44	57.67	5.44	
70	M	68.3	2.78	57.67	152.11	
75	M	68.3	44.44	57.67	300.44	
			220.67		903.33	



วิธีคิด cost : R^2 (ในที่นี้ยิ่งเยอะยิ่งดี)

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$$

SSE (SSres) = sum((Actual – predict)^2)
= sum((55-47)^2 , (45-47)^2 , (41-47)^2 , (60-68.3)^2 , (70-68.3)^2 , (75-68.3)^2)
= 220.67

SST (SStot) = sum((Actual – mean(Actual))^2)
mean(actual) = average(55 ,45 ,41 , 60 , 70 , 75) = 57.67
SST (SStot) = sum((55-57.67)^2 , (45-57.67)^2 , (41-57.67)^2 , (60-57.67)^2 , (70-57.67)^2 , (75-57.67)^2)
= 903.33

R_squared = 1 – (SSE/SST)
= 1- (220.67/903.33)
= **0.756**

Tree Regression (Decision Tree step)

Decision Tree Regression step

1. หา cost ทั้งหมดที่ตัว Root ก่อนเริ่ม split (เช่น หา R squared ก่อนเริ่ม split)
2. Split โดยใช้ ทุก criteria (feature and data cut off) และหา cost ของ ทุกการ split แล้วเลือก criteria split ที่ให้ Rsquared เยอะสุด
3. Split ไปเรื่อยๆ จนกว่าจะ ถึง criteria ที่กำหนด (เช่น จำนวน data point ให้ leaf node < 5 หรือ split ต่อไปแล้ว R_squared เพิ่มน้อยมาก)

NO SPLIT

weight	gender	height	prediction (all average)	sum of squared error	sum of squared total	rsquared
55	F	165	57.67	7.11	7.11	**0.000**
45	F	159	57.67	160.44	160.44	
41	F	151	57.67	277.78	277.78	
60	M	165	57.67	5.44	5.44	
70	M	171	57.67	152.11	152.11	
75	M	180	57.67	300.44	300.44	
				903.33	**903.33**	

R_squared เพิ่มขึ้น
-> SPLIT ได้!!!

SPLIT BY GENDER

weight	gender	height	prediction (split by gender)	sum of squared error	sum of squared total	rsquared
55	F	165	47	64	7.11	**0.756**
45	F	159	47	4	160.44	
41	F	151	47	36	277.78	
60	M	165	68.3	69.44	5.44	
70	M	171	68.3	2.78	152.11	
75	M	180	68.3	44.44	300.44	
				220.67	**903.33**	

Tree Regression (Decision Tree step)

NO SPLIT

weight	gender	height	prediction (all average)	sum of squared error	sum of squared total	rsquared
55	F	165	57.67	7.11	7.11	**0.000**
45	F	159	57.67	160.44	160.44	
41	F	151	57.67	277.78	277.78	
60	M	165	57.67	5.44	5.44	
70	M	171	57.67	152.11	152.11	
75	M	180	57.67	300.44	300.44	
				903.33	**903.33**	

SPLIT BY HEIGHT >= 162

weight	gender	height	prediction (split 162)	sum of squared error	sum of squared total	rsquared
41	F	151	43	4.00	277.78	**0.714**
45	F	159	43	4.00	160.44	
55	F	165	65	100.00	7.11	
60	M	165	65	25.00	5.44	
70	M	171	65	25.00	152.11	
75	M	180	65	100.00	300.44	
				258.00	**903.33**	

SPLIT BY GENDER

weight	gender	height	prediction (split by gender)	sum of squared error	sum of squared total	rsquared
55	F	165	47	64	7.11	**0.756**
45	F	159	47	4	160.44	
41	F	151	47	36	277.78	
60	M	165	68.3	69.44	5.44	
70	M	171	68.3	2.78	152.11	
75	M	180	68.3	44.44	300.44	
				220.67	**903.33**	

SPLIT BY HEIGHT >= 168

weight	gender	height	prediction (split 168)	sum of squared error	sum of squared total	rsquared
41	F	151	50.25	50.25	277.78	**0.745**
45	F	159	50.25	50.25	160.44	
55	F	165	50.25	22.56	7.11	
60	M	165	50.25	95.06	5.44	
70	M	171	72.5	6.25	152.11	
75	M	180	72.5	6.25	300.44	
				230.63	**903.33**	

SPLIT BY HEIGHT >= 155

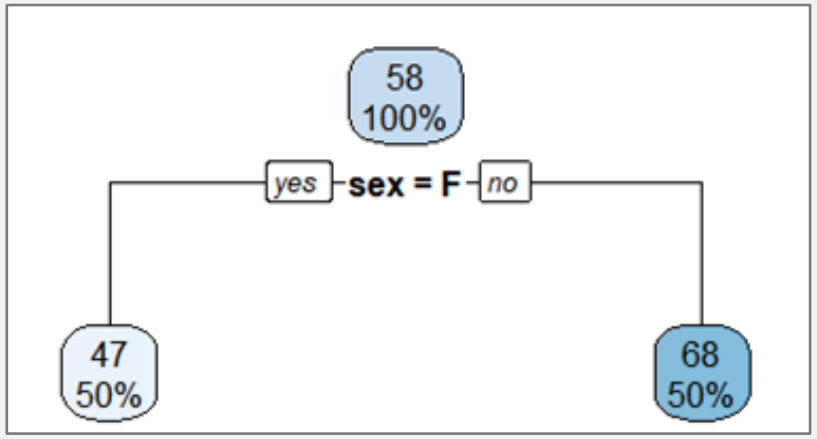
weight	gender	height	prediction (split 155)	sum of squared error	sum of squared total	rsquared
41	F	151	41	0.00	277.78	**0.369**
45	F	159	61	256.00	160.44	
55	F	165	61	36.00	7.11	
60	M	165	61	1.00	5.44	
70	M	171	61	81.00	152.11	
75	M	180	61	196.00	300.44	
				570.00	**903.33**	

SPLIT BY HEIGHT >= 175

weight	gender	height	prediction (split 175)	sum of squared error	sum of squared total	rsquared
41	F	151	54.2	174.24	277.78	**0.399**
45	F	159	54.2	84.64	160.44	
55	F	165	54.2	0.64	7.11	
60	M	165	54.2	33.64	5.44	
70	M	171	54.2	249.64	152.11	
75	M	180	75	0.00	300.44	
				542.80	**903.33**	

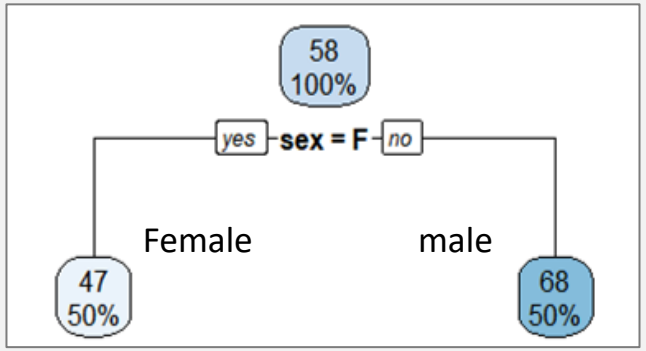
Tree Regression (Decision Tree step)

Criteria 1st root	Rsquared
No split	**0**
Split by gender ✓	**0.756**
Split by height (155)	**0.369**
Split by height (162)	**0.714**
Split by height (168)	**0.745**
Split by height (175)	**0.399**



หลังจาก gender จะ ใช้อะไรในการ split ??

Tree Regression (Decision Tree step)



SPLIT BY HEIGHT >= 155

weight	gender	height	Prediction (split height 155)	sum of squared error	sum of squared total	rsquared
41	F	151	41	0	277.78	**0.888**
45	F	159	50	25	160.44	
55	F	165	50	25	7.11	
				50	**445.33**	

SPLIT BY HEIGHT >= 162

weight	gender	height	Prediction (split height 162)	sum of squared error	sum of squared total	rsquared
41	F	151	43	4	277.78	**0.982**
45	F	159	43	4	160.44	
55	F	165	55	0	7.11	
				8	**445.33**	

SPLIT BY HEIGHT >= 168

weight	gender	height	Prediction (split height 168)	sum of squared error	sum of squared total	rsquared
60	M	165	60	0	5.44	**0.973**
70	M	171	72.5	6.25	152.11	
75	M	180	72.5	6.25	300.44	
				12.5	**458**	

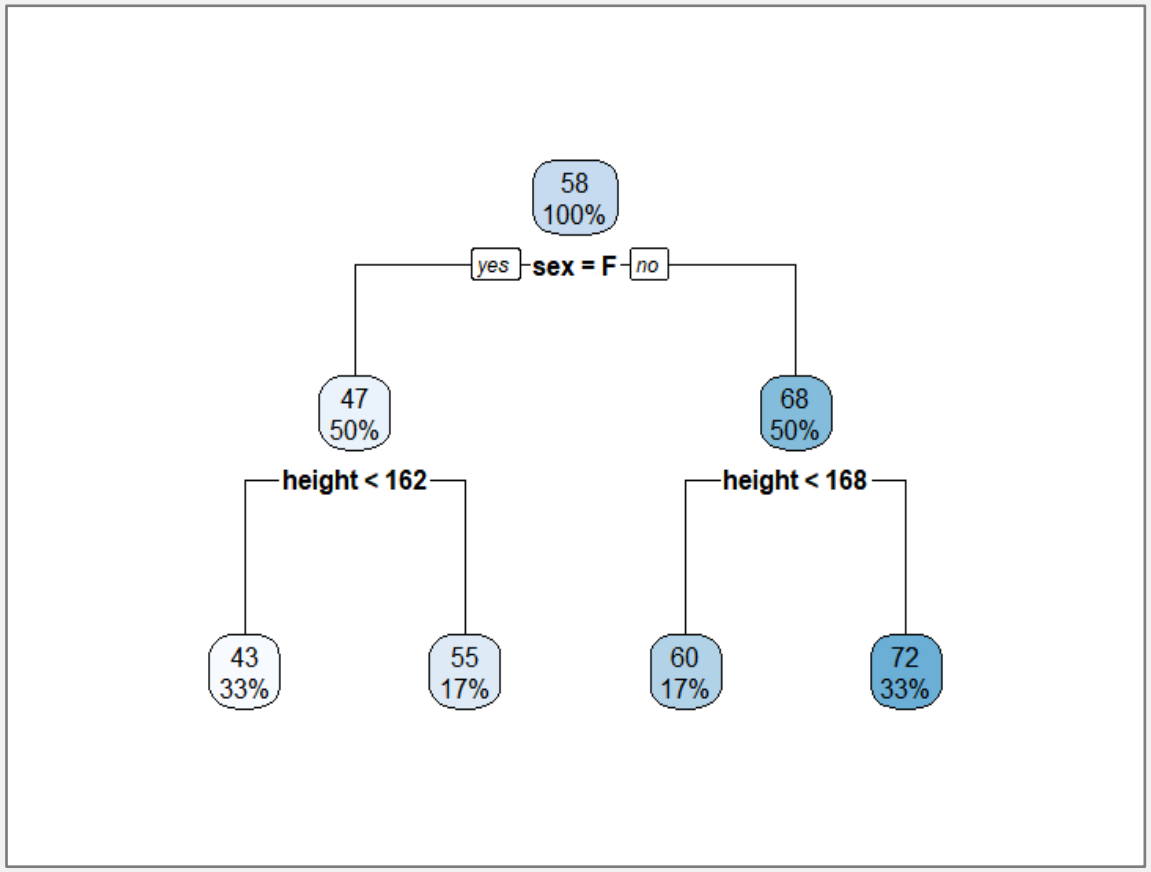
SPLIT BY HEIGHT >= 175.5

weight	gender	height	Prediction (split height 175.5)	sum of squared error	sum of squared total	rsquared
60	M	165	65	25	5.44	**0.891**
70	M	171	65	25	152.11	
75	M	180	75	0	300.44	
				50	**458**	

Tree Regression (Decision Tree step)

Criteria 1st root	Rsquared
After gender split	**0.756**
Prediction (split height 155)	**0.888**
Prediction (split height 162) ✓	**0.982**

Criteria 1st root	Rsquared
After gender split	**0.756**
Prediction (split height 168) ✓	**0.973**
Prediction (split height 175.5)	**0.891**



ใช้ Split โดย height < 162 สำหรับ gender female
และ height < 168 สำหรับ gender male

Tree Regression (Decision Tree step)

Best Split คือ

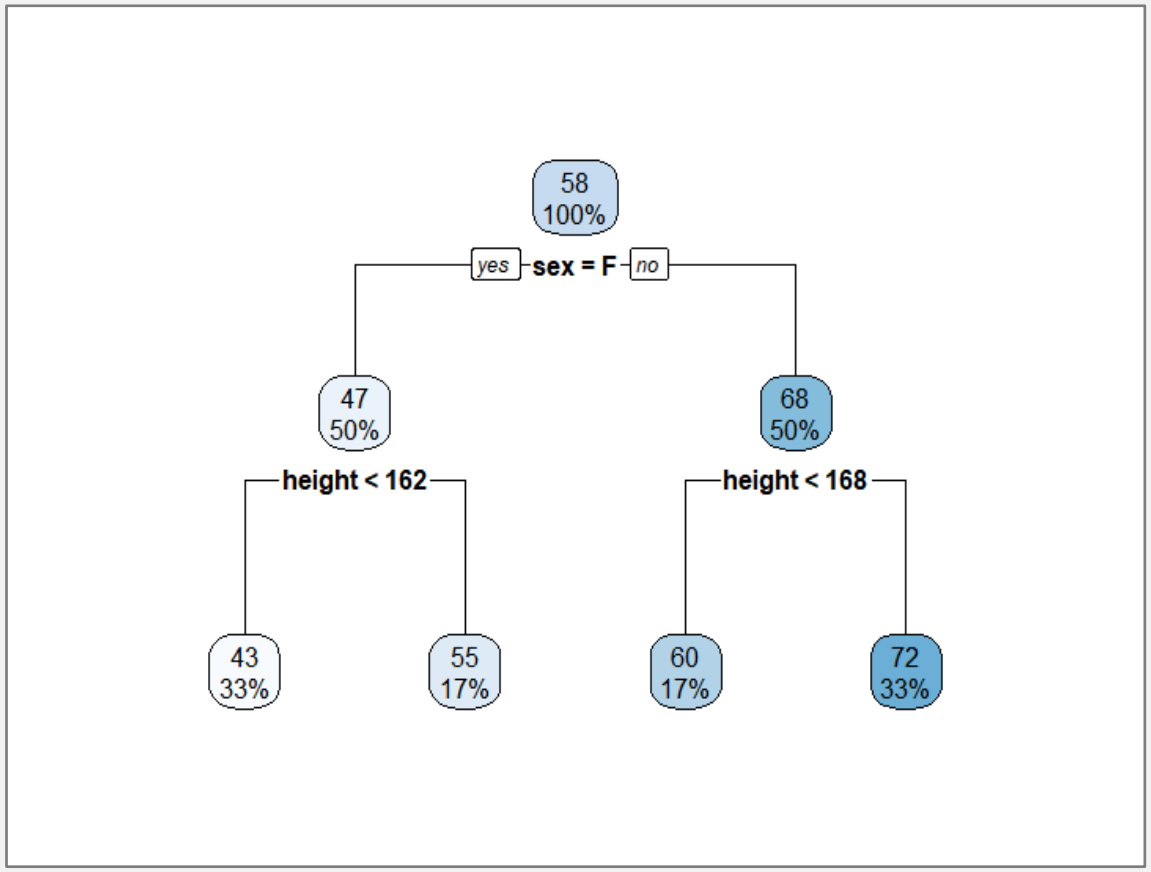
1 : Split by gender

2.1 : Split by **height** 162 สำหรับ sex = F

2.2 : Split by **height** 168 สำหรับ sex = M

Final R squared = 0.977

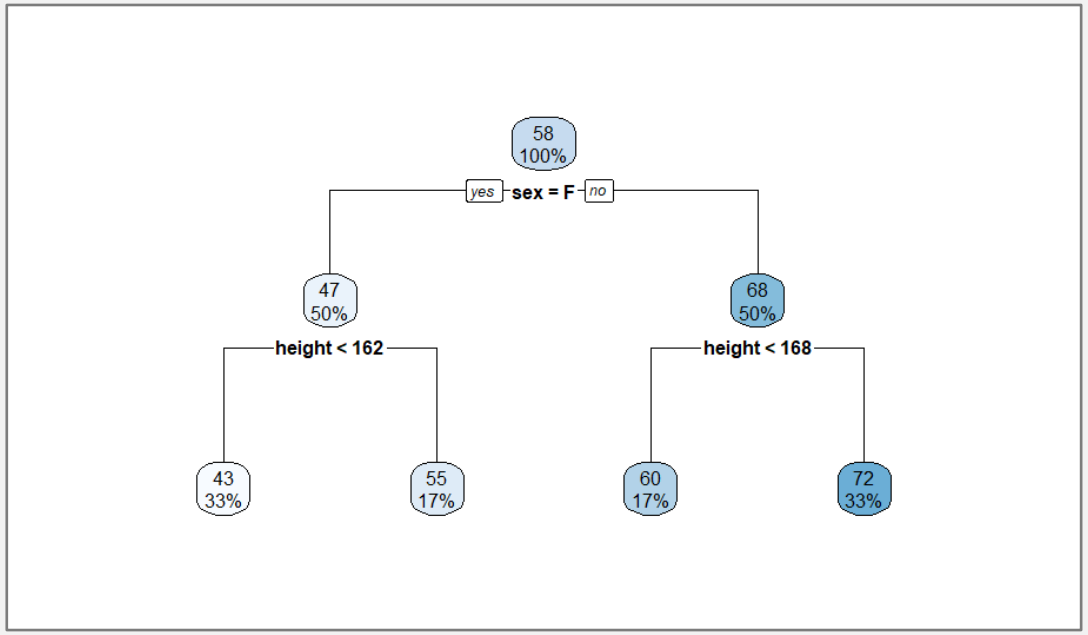
weight	gender	height	prediction (split height 162 & 168)	sum of squared error	sum of squared total	rsquared
41	F	151	43	4	277.78	**0.977**
45	F	159	43	4	160.44	
55	F	165	55	0	7.11	
60	M	165	60	0	5.44	
70	M	171	72.5	6.25	152.11	
75	M	180	72.5	6.25	300.44	
				20.50	903.33	



Note : decision Tree จะ split ไปเรื่อยๆ จนกว่าจะมีค่า Prediction = Data Point หากไม่มีการ Control Overfit เช่น Pruning

Tree Regression (Decision Tree Code)

```
library(rpart)
library(rpart.plot)

rpart_con <-  rpart.control(minsplit = 3, cp = 0.01, maxdepth = 30)

tree_model <- rpart(weight ~ .,weight_data , control = rpart_con)
rpart.plot(tree_model)
```



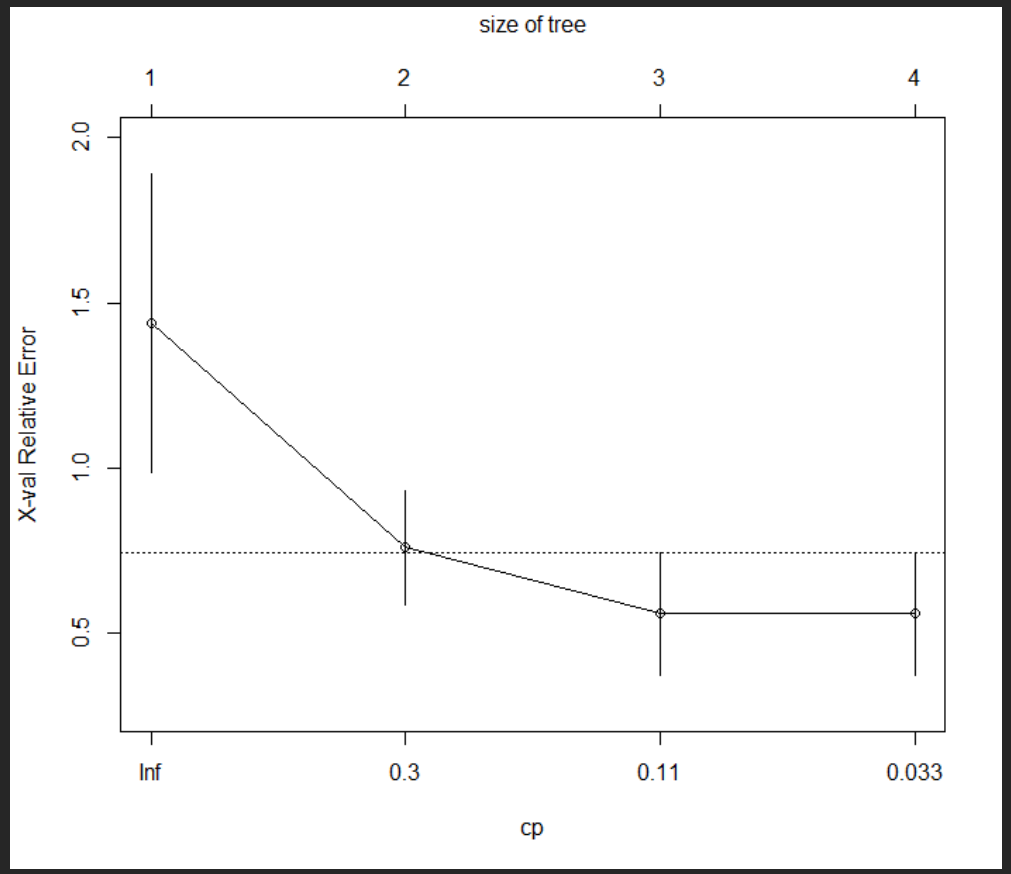
Tree Regression (Decision Tree Code)

```
library(rpart)
library(rpart.plot)

rpart_con <-  rpart.control(minsplit = 3, cp = 0.01, maxdepth = 30)

tree_model <- rpart(weight ~ .,weight_data , control = rpart_con)
rpart.plot(tree_model)

printcp(tree_model)
plotcp(tree_model)
summary(tree_model)
par(mfrow=c(1,2))
rsq.rpart(tree_model)
```

Tree Regression (Decision Tree Code)

```
printcp(tree_model)

Regression tree:
rpart(formula = weight ~ ., data = weight_data, control
= rpart_con)

Variables actually used in tree construction:
[1] height sex

Root node error: 903.33/6 = 150.56

n= 6

       CP nsplit rel error  xerror    xstd
1 0.75572      0  1.000000 1.44000 0.45196
2 0.11531      1  0.244280 0.75873 0.17164
3 0.10627      2  0.128967 0.55766 0.18587
4 0.01000      3  0.022694 0.55766 0.18587
```

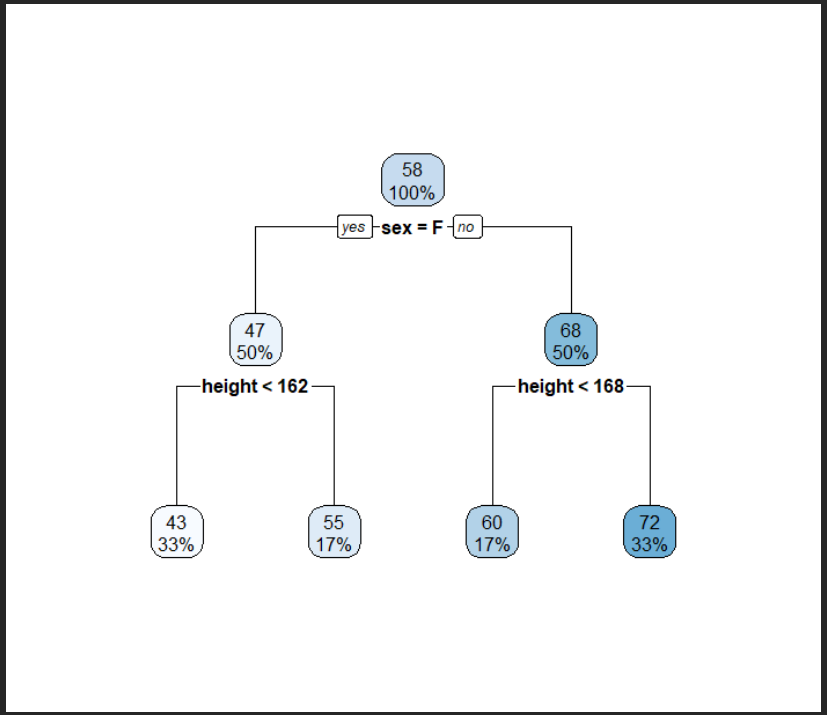
```
plotcp(tree_model)
```



Tree Regression (Decision Tree Code)

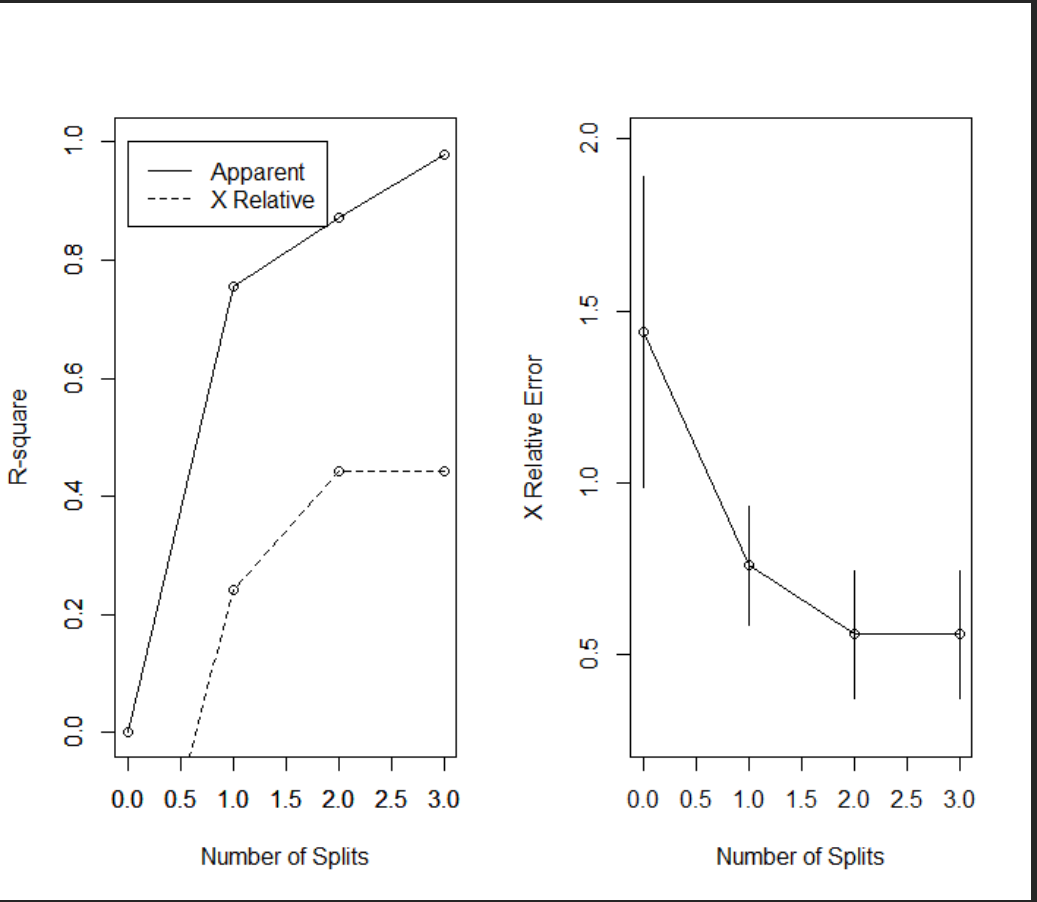
```
summary(tree_model)
Call:
rpart(formula = weight ~ ., data = weight_data, control = rpart_con)
  n= 6

         CP nsplit  rel error    xerror      xstd
1 0.7557196      0 1.00000000 1.4400000 0.4519646
2 0.1153137      1 0.24428044 0.7587331 0.1716398
3 0.1062731      2 0.12896679 0.5576568 0.1858664
4 0.0100000      3 0.02269373 0.5576568 0.1858664

Variable importance
   sex height
    51     49

Node number 1: 6 observations,    complexity param=0.7557196
  mean=57.66667, MSE=150.5556
  left son=2 (3 obs) right son=3 (3 obs)
  Primary splits:
      sex    splits as  LR,      improve=0.7557196, (0 missing)
      height < 168 to the left,  improve=0.7307196, (0 missing)
  Surrogate splits:
      height < 162 to the left,  agree=0.833, adj=0.667, (0 split)

Node number 2: 3 observations,    complexity param=0.1062731
  mean=47, MSE=34.66667
  left son=4 (2 obs) right son=5 (1 obs)
  Primary splits:
      height < 162 to the left,  improve=0.9230769, (0 missing)

Node number 3: 3 observations,    complexity param=0.1153137
  mean=68.33333, MSE=38.88889
  left son=6 (1 obs) right son=7 (2 obs)
  Primary splits:
      height < 168 to the left,  improve=0.8928571, (0 missing)

Node number 4: 2 observations
  mean=43, MSE=4

Node number 5: 1 observations
  mean=55, MSE=0

Node number 6: 1 observations
  mean=60, MSE=0

Node number 7: 2 observations
  mean=72.5, MSE=6.25
```

```
rpart.plot(tree_model)
```



Tree Regression (Decision Tree Code)

```
par(mfrow=c(1,2))
rsq.rpart(tree_model)

Regression tree:
rpart(formula = weight ~ ., data = weight_data,
                            control = rpart_con)

Variables actually used in tree construction:
[1] height sex

Root node error: 903.33/6 = 150.56

n= 6

       CP nsplit rel error  xerror    xstd
1 0.75572      0  1.000000 1.44000 0.45196
2 0.11531      1  0.244280 0.75873 0.17164
3 0.10627      2  0.128967 0.55766 0.18587
4 0.01000      3  0.022694 0.55766 0.18587
```



Tree Regression (Decision Tree Code Caret Version)

Caret Version

```
tree_model_caret <- train(weight~.,
                          data = weight_data,
                          method = "rpart")
tree_model_caret
```

Tree Regression (Decision Tree overfitting Tree)

หาก split เยอะมากเกินไป (ต้นไม้ที่ลึกมาก) อาจจะทำให้เกิดปัญหา **overfit ได้**
วิธีทำให้ ต้นไม้ split น้อยลง

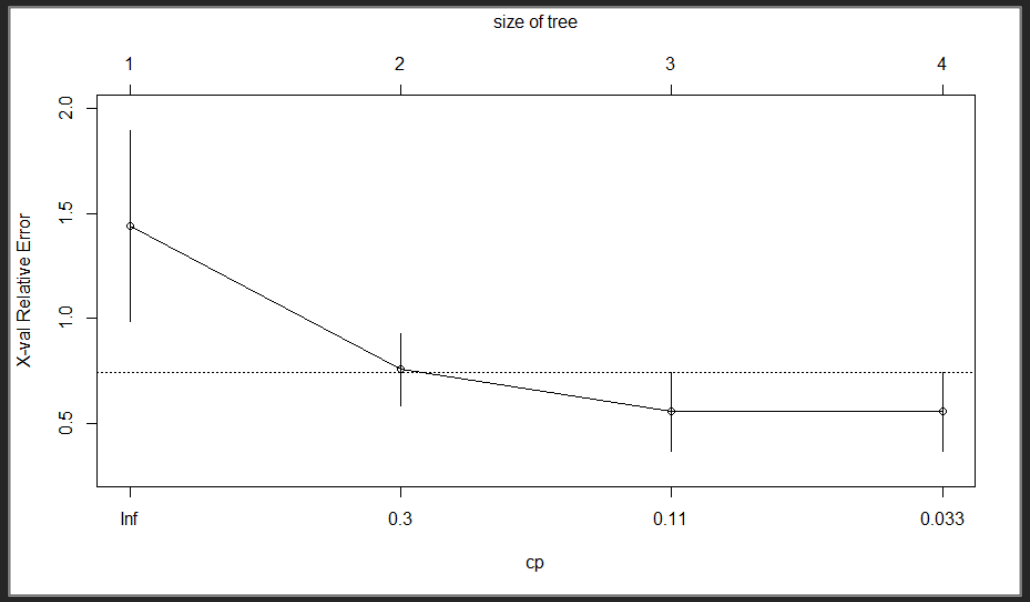
- Tune parameter (minsplit , minbucket ,maxdepth ,cp ....)
- **ตัดกิ่งต้นไม้ Pruning Decision Tree**

Recall : overfit คือ ปัญหาที่ model fit ข้อมูล Training Data ดีมากเกินไป (error น้อย) แต่พอนำ model ไป predict Testing Data error เยอะ

Tree Regression (Decision Tree Pruning)

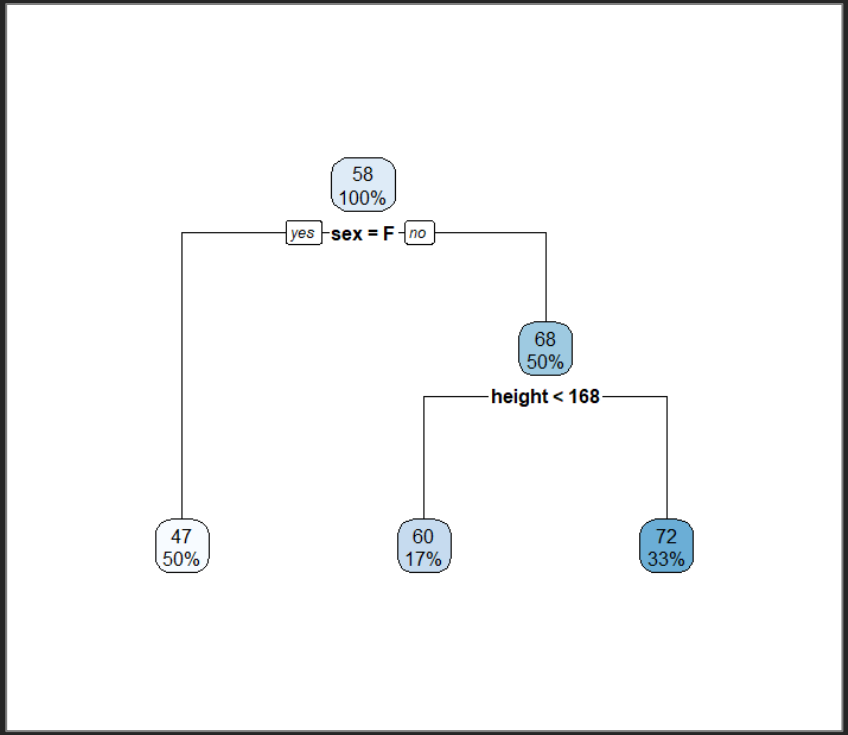
เราสามารถลดขนาดของต้นไม้ได้ โดยใช้การ **Pruning** (การตัดกิ่ง) โดยใช้ **cp parameter** เป็นตัว set (จะตัดกิ่งไปเรื่อยๆจนกว่า r squared จะลดมากกว่าที่กำหนดไว้ [cp])

```
plotcp(tree_model)
```




```
tree_fit<- prune(tree_model , cp = 0.11)
tree_fit
```

```
rpart.plot(tree_fit)
```



Tree Regression (Decision Tree Pros/Cons)

Pros

1. เป็น Model ที่**สร้างง่าย** และ**อธิบายง่าย**
2. เป็น Model **ที่ไม่ต้อง Prep data เยอะ** เช่น ไม่สนใจ scaling , ไม่สนใจ distribution , ไม่สนใจ categorical data
3. เป็น Model **ที่ไม่ต้อง defined functional form** ไว้ก่อนแบบ regression เช่น y = b0 + b1x
4. สำหรับนำไปใช้ใน **feature selection** ได้

Cons

1. โดยส่วนมาก **ใช้เวลา Train มากกว่า linear regression**
2. การ predict บางทีจะไป **กองอยู่ที่ค่าๆหนึ่ง**
3. การ Predict **ค่อนข้างจะไม่แม่น** เมื่อเทียบกับ Model อื่นๆ

Random Forest Regression

Random Forest Regression (Fundamental)

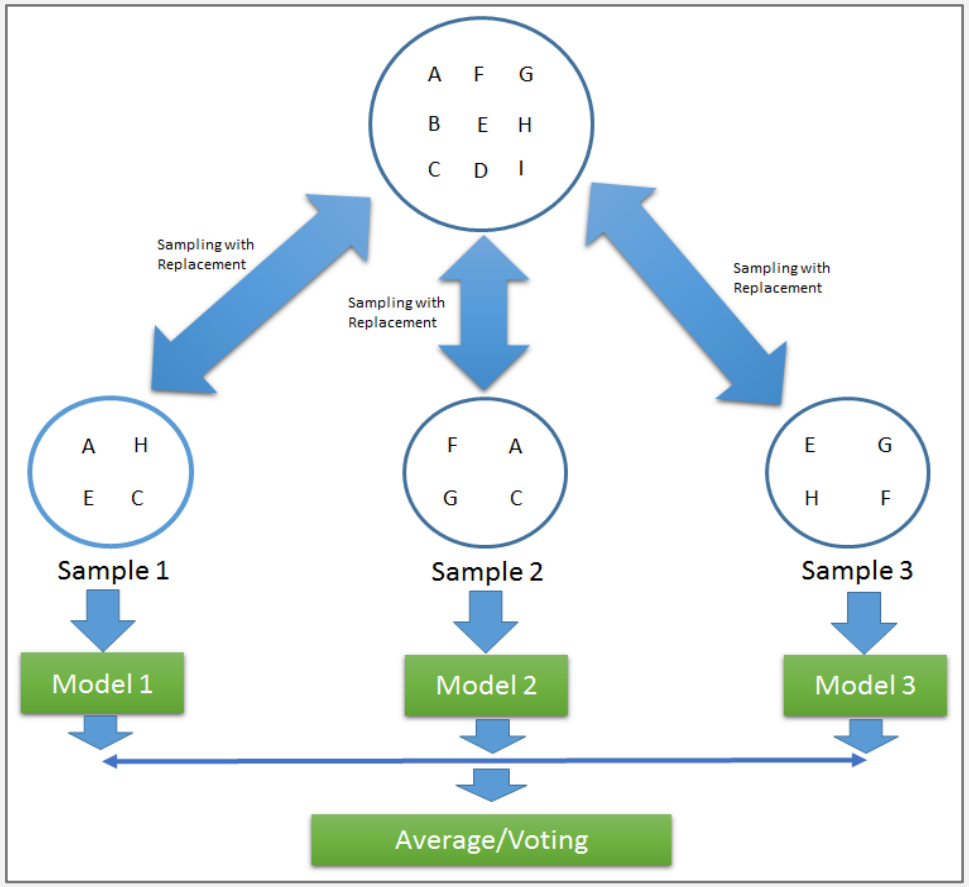
Decision Tree Regression

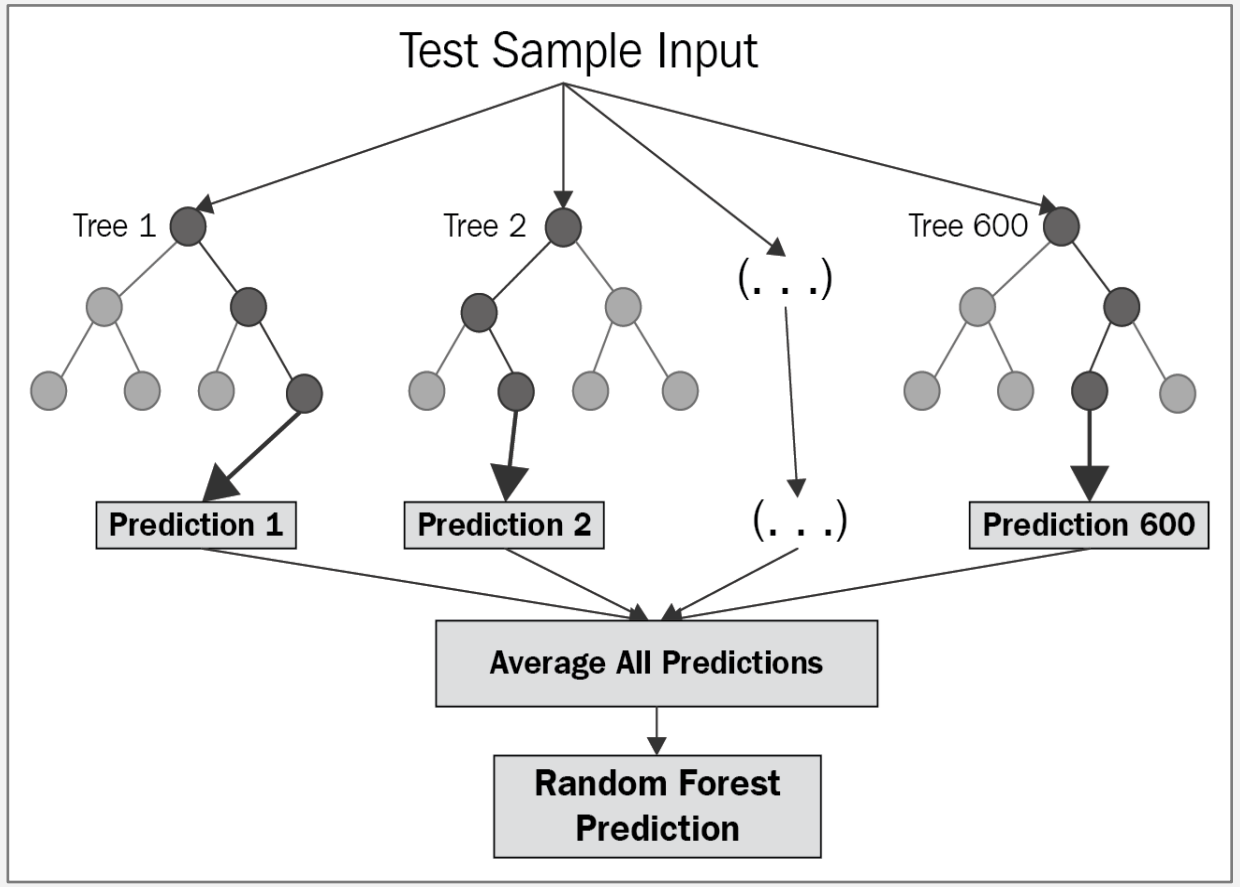
- ศร้าง Decision Tree **ต้นไม้ ต้นเดียว** ผล prediction เลยมาจากต้นเดียว
- ใช้ **ทุก Feature** เป็น ตัวเลือกในการ split
- ใช้ **ทุก Data Point** ในการสร้าง ต้นไม้

Random Forest Regression

- ศร้าง Decision Tree **ต้นไม้ หลายต้น** ผล prediction เลยมาจากการ **average ผลของแต่ละต้น**
- แต่ละต้น จะ **random subset ของ Feature** เป็นตัวเลือกในการ split เช่นมี 8 feature (อาจจะ random มาใช้ต่อการ split แค่ 3 feature เป็นตัวเลือก)
- ใช้ **sample ของ Data Point** ในการสร้าง ต้นไม้แต่ละต้น (sample with replacement)

Random Forest Regression (Fundamental)





https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781788830577/3/ch03lvl1sec34/bagging

Random Forest Regression (Fundamental)

สมมุติ เราต้องการทำ Model Predict Weight

Target (Y) : Weight

Feature (X) : Gender , height , blood , age

ID	weight	gender	height	blood	age
1	55	F	165	B	40
2	45	F	159	AB	34
3	41	F	151	B	21
4	60	M	165	B	26
5	70	M	171	A	37
6	75	M	180	AB	54
7	68	M	175	B	22
8	50	F	170	A	27
9	48	F	164	AB	49
10	64	M	166	B	35

Random Forest Regression (Fundamental)

Whole dataset

ID	weight	gender	height	blood	age
1	55	F	165	B	40
2	45	F	159	AB	34
3	41	F	151	B	21
4	60	M	165	B	26
5	70	M	171	A	37
6	75	M	180	AB	54
7	68	M	175	B	22
8	50	F	170	A	27
9	48	F	164	AB	49
10	64	M	166	B	35

1st Tree Dataset

ID	weight	gender	height	blood	age
1	55	F	165	B	40
2	45	F	159	AB	34
4	60	M	165	B	26
6	75	M	180	AB	54
9	48	F	164	AB	49
9	48	F	164	AB	49

2nd Tree Dataset

ID	weight	gender	height	blood	age
1	55	F	165	B	40
4	60	M	165	B	26
4	60	M	165	B	26
5	70	M	171	A	37
7	68	M	175	B	22
8	50	F	170	A	27

3rd Tree Dataset

ID	weight	gender	height	blood	age
2	45	F	159	AB	34
3	41	F	151	B	21
6	75	M	180	AB	54
7	68	M	175	B	22
10	64	M	166	B	35
10	64	M	166	B	35

Random Forest Regression (Fundamental)

Whole dataset

ID	weight	gender	height	blood	age
1	55	F	165	B	40
2	45	F	159	AB	34
3	41	F	151	B	21
4	60	M	165	B	26
5	70	M	171	A	37
6	75	M	180	AB	54
7	68	M	175	B	22
8	50	F	170	A	27
9	48	F	164	AB	49
10	64	M	166	B	35

1st tree	1st split			2nd split	
set of feature	gender	blood		height	age

2nd tree	1st split			2nd split	
set of feature	blood	age		age	gender

3rd tree	1st split			2nd split	
set of feature	gender	height		blood	age

1st Tree Dataset

ID	weight	gender	height	blood	age
1	55	F	165	B	40
2	45	F	159	AB	34
4	60	M	165	B	26
6	75	M	180	AB	54
9	48	F	164	AB	49
9	48	F	164	AB	49

2nd Tree Dataset

ID	weight	gender	height	blood	age
1	55	F	165	B	40
4	60	M	165	B	26
4	60	M	165	B	26
5	70	M	171	A	37
7	68	M	175	B	22
8	50	F	170	A	27

3rd Tree Dataset

ID	weight	gender	height	blood	age
2	45	F	159	AB	34
3	41	F	151	B	21
6	75	M	180	AB	54
7	68	M	175	B	22
10	64	M	166	B	35
10	64	M	166	B	35

Random Forest Regression (Fundamental)

Whole dataset

ID	weight	gender	height	blood	age
1	55	F	165	B	40
2	45	F	159	AB	34
3	41	F	151	B	21
4	60	M	165	B	26
5	70	M	171	A	37
6	75	M	180	AB	54
7	68	M	175	B	22
8	50	F	170	A	27
9	48	F	164	AB	49
10	64	M	166	B	35

1st tree	1st split			2nd split	
set of feature	gender	blood		height	age

2nd tree	1st split			2nd split	
set of feature	blood	age		age	gender

3rd tree	1st split			2nd split	
set of feature	gender	height		blood	age

1st Tree Dataset

ID	weight	gender	height	blood	age
1	55	F	165	B	40
2	45	F	159	AB	34
4	60	M	165	B	26
6	75	M	180	AB	54
9	48	F	164	AB	49
9	48	F	164	AB	49

2nd Tree Dataset

ID	weight	gender	height	blood	age
1	55	F	165	B	40
4	60	M	165	B	26
4	60	M	165	B	26
5	70	M	171	A	37
7	68	M	175	B	22
8	50	F	170	A	27

3rd Tree Dataset

ID	weight	gender	height	blood	age
2	45	F	159	AB	34
3	41	F	151	B	21
6	75	M	180	AB	54
7	68	M	175	B	22
10	64	M	166	B	35
10	64	M	166	B	35

Random Forest Regression (Fundamental)

Whole dataset

ID	weight	gender	height	blood	age
1	55	F	165	B	40
2	45	F	159	AB	34
3	41	F	151	B	21
4	60	M	165	B	26
5	70	M	171	A	37
6	75	M	180	AB	54
7	68	M	175	B	22
8	50	F	170	A	27
9	48	F	164	AB	49
10	64	M	166	B	35

1st tree	1st split			2nd split	
set of feature	gender	blood		height	age

2nd tree	1st split			2nd split	
set of feature	blood	age		age	gender

3rd tree	1st split			2nd split	
set of feature	gender	height		blood	age

1st Tree Dataset

ID	weight	gender	height	blood	age
1	55	F	165	B	40
2	45	F	159	AB	34
4	60	M	165	B	26
6	75	M	180	AB	54
9	48	F	164	AB	49
9	48	F	164	AB	49

2nd Tree Dataset

ID	weight	gender	height	blood	age
1	55	F	165	B	40
4	60	M	165	B	26
4	60	M	165	B	26
5	70	M	171	A	37
7	68	M	175	B	22
8	50	F	170	A	27

3rd Tree Dataset

ID	weight	gender	height	blood	age
2	45	F	159	AB	34
3	41	F	151	B	21
6	75	M	180	AB	54
7	68	M	175	B	22
10	64	M	166	B	35
10	64	M	166	B	35

Random Forest Regression (Fundamental)

1st Tree

Predicted : 55

2nd Tree

Predicted : 60

3rd Tree

Predicted : 65

Average Prediction

(55+60+65)/3

= 60

Random Forest Regression (R Code)

```
library(ranger)
randomforest_model <- ranger(formula = medv ~.,
        data = BostonHousing,
        num.tree = 21,
        mtry = 5,
        importance = "permutation")
randomforest_model

Ranger result

Call:
 ranger(formula = medv ~ ., data = BostonHousing, num.tree = 21,      mtry = 5, importance = "permutation")

Type:                             Regression
Number of trees:                  21
Sample size:                      506
Number of independent variables:  13
Mtry:                             5
Target node size:                 5
Variable importance mode:         permutation
Splitrule:                        variance
OOB prediction error (MSE):       11.99563
R squared (OOB):                  0.8581854
```

Argument ที่น่าสนใจ

mtry (จำนวน feature ที่จะใช้ ในการ split แต่ละครั้ง)

min.node.size (จำนวน data point ขั้นต่ำใน node สุดท้าย [terminal leaf])

sample.fraction (สัดส่วนของ dataset ที่ จะถูกนำไปสร้าง ต้นไม้แต่ละต้น)

num.tree (จำนวน ต้นไม้ทั้งหมด)

Random Forest Regression (R Code)

```
df_importance <- data.frame(randomforest_model$variable.importance)
names(df_importance) <- "importance_score"
df_importance[order(df_importance$importance_score , decreasing = TRUE),,drop = FALSE]
        importance_score
lstat         68.2198104
rm            37.5702637
crim          10.4620399
nox            5.6233907
indus          5.4479809
ptratio        4.9878843
tax            4.5365892
dis            3.9570106
age            3.1985715
b              1.4325235
rad            0.7373876
zn             0.6361465
chas          -1.2144359
```

Random Forest Regression (R Code Caret)

Caret version

```
library(ranger)
library(caret)
set.seed(639)

randomforest_model_caret <- train(medv~.,
                    data = BostonHousing,
                    importance = "permutation",
                    method = "ranger")
randomforest_model_caret
```

Tree Regression (Random Forest Pros/ Cons)

Pros

1. เป็น Model ที่ **Prediction ค่อนข้างแม่น** เมื่อเทียบกับ Model หลายๆตัว
2. สำหรับนำไปใช้ใน **feature selection** ได้
3. ตัว Model **แก้ ปัญหา Overfit ไปในตัว** เพราะมีการแบ่ง ต้นไม้ + subset feature สำหรับ split + sampling data สำหรับสร้างต้นไม้

Cons

1. **อธิบายผลยากกว่า** Decision Tree
2. ใช้เวลา **Train + Predict นาน** (แต่การรัน parallel ช่วยได้)

Gradient Boosting Tree Regression

Gradient Boosting Regression (Fundamental)

Random Forest Regression

VS

Gradient Boosting Tree Regression

- สร้าง Decision Tree ต้นไม้ **หลายต้น แบบ parallel** (แต่ละต้น independent ต่อกัน) ผล prediction เลยมาจากการ average ผลของแต่ละต้น
- ใช้ **sample ของ Data Point** ในการสร้าง ต้นไม้แต่ละต้น (sample with replacement)

- ศร้าง Decision Tree **ต้นไม้ ต้นหลายต้น แบบ sequential** (แต่ละต้น dependent ต่อกัน) ผล prediction มาจากการ average ข้อมูลตอนแรก + ด้วย adjusted error ไปเรื่อยๆ ตามลำดับต้นไม้ที่เราสร้างขึ้นมา
- **ใช้ทุก Data Point** ในการสร้าง ต้นไม้ -- (**Gradient Boosting Tree**)
- **ใช้ Sample of Data Point** ในการสร้างต้นไม้ -- (**Stochastic Gradient Boosting Tree**)
- Note: Stochastic Gradient Boosting Tree มี research ที่บอกว่าการ sub sample ทำให้เกิดผล prediction ใน test set ดีกว่า

Gradient Boosting Regression (Step)

Assume ว่าเราทำต้นไม้ 3 ต้น sequentially

weight	gender	height
41	F	151
45	F	159
55	F	165
60	M	165
70	M	171
75	M	180

Weight prediction ครั้งแรกคือการ นำ weight ไป average (41+45+55+60+70+75)/6 = **57.667**

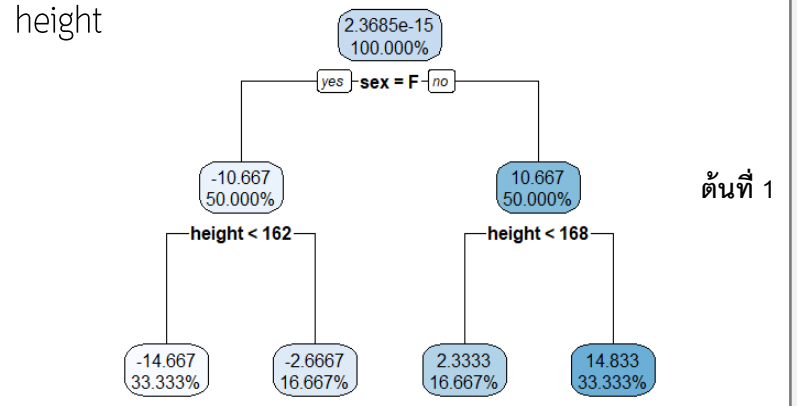
weight	sex	height	**average weight**
55	F	165	**57.667**
45	F	159	**57.667**
41	F	151	**57.667**
60	M	165	**57.667**
70	M	171	**57.667**
75	M	180	**57.667**

Gradient Boosting Regression (Step)

Calculate error (error_0) = Actual weight – Average weight

weight	sex	height	average weight	error_0
55	F	165	57.667	**-2.667**
45	F	159	57.667	**-12.667**
41	F	151	57.667	**-16.667**
60	M	165	57.667	**2.333**
70	M	171	57.667	**12.333**
75	M	180	57.667	**17.333**

นำ error_0 ที่ได้มา ไปทำ Decision Tree model โดย Y variable คือ error_0 , Feature คือ sex และ height



ต้นที่ 1

weight	sex	height	average weight	error_0	error_pred1
55	F	165	57.667	-2.667	**-2.667**
45	F	159	57.667	-12.667	**-14.667**
41	F	151	57.667	-16.667	**-14.667**
60	M	165	57.667	2.333	**2.333**
70	M	171	57.667	12.333	**14.833**
75	M	180	57.667	17.333	**14.833**

Gradient Boosting Regression (Step)

นำ error ที่ได้จาก decision tree (error_pred1) ไปคูณ การ learning rate (shrinkage rate/ alpha) assume ว่า learning rate = 0.1

error_pred1_alpha = error_pred1 * 0.1

weight	sex	height	average weight	error_0	error_pred1	**error_pred1_alpha**
55	F	165	57.667	-2.667	-2.667	**-0.267**
45	F	159	57.667	-12.667	-14.667	**-1.467**
41	F	151	57.667	-16.667	-14.667	**-1.467**
60	M	165	57.667	2.333	2.333	**0.233**
70	M	171	57.667	12.333	14.833	**1.483**
75	M	180	57.667	17.333	14.833	**1.483**

นำ average weight + error_pred1_alpha จะได้ weight_1 เป็นค่า prediction จากต้นไม้ต้นที่ 1

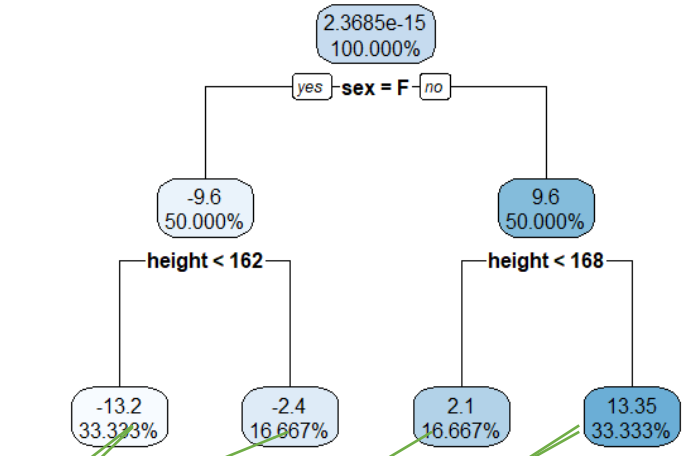
weight	sex	height	average weight	error_0	error_pred1	error_pred1_alpha	**weight_1**
55	F	165	57.667	-2.667	-2.667	-0.267	**57.4**
45	F	159	57.667	-12.667	-14.667	-1.467	**56.2**
41	F	151	57.667	-16.667	-14.667	-1.467	**56.2**
60	M	165	57.667	2.333	2.333	0.233	**57.9**
70	M	171	57.667	12.333	14.833	1.483	**59.15**
75	M	180	57.667	17.333	14.833	1.483	**59.15**

Gradient Boosting Regression (Step)

เราเริ่มทำต้นที่ 2 โดยการหา error ที่ model predicted ผลาด (error_1)
error_1 = weight - weight_1

weight	sex	height	weight_1	error_1
55	F	165	57.4	-2.4
45	F	159	56.2	-11.2
41	F	151	56.2	-15.2
60	M	165	57.9	2.1
70	M	171	59.15	10.85
75	M	180	59.15	15.85

นำ error_1 ที่ได้มา ไปทำ Decision Tree model
โดย Y variable คือ error_1 , Feature คือ sex และ height



ต้นที่ 2

weight	sex	height	weight_1	error_1	error_pred2
55	F	165	57.4	-2.4	-2.400
45	F	159	56.2	-11.2	-13.200
41	F	151	56.2	-15.2	-13.200
60	M	165	57.9	2.1	2.100
70	M	171	59.15	10.85	13.350
75	M	180	59.15	15.85	13.350

Gradient Boosting Regression (Step)

นำ error ที่ได้จาก decision tree (error_pred2) ไปคูณ การ learning rate (shrinkage rate/ alpha) assume ว่า learning rate = 0.1
error_pred2_alpha = error_pred2 * 0.1

weight	sex	height	weight_1	error_1	error_pred2	error_pred2_alpha
55	F	165	57.4	-2.4	-2.400	-0.24
45	F	159	56.2	-11.2	-13.200	-1.32
41	F	151	56.2	-15.2	-13.200	-1.32
60	M	165	57.9	2.1	2.100	0.21
70	M	171	59.15	10.85	13.350	1.335
75	M	180	59.15	15.85	13.350	1.335

Weight_2 = average weight + error_pred1_alpha + error_pred2_alpha
(weight_2 เป็นค่า prediction จากต้นไม้ต้นที่ 2)

weight	sex	height	average weight	error_pred1_alpha	error_pred2_alpha	weight_2
55	F	165	**57.667**	**-0.267**	**-0.24**	**57.16**
45	F	159	**57.667**	**-1.467**	**-1.32**	**54.88**
41	F	151	**57.667**	**-1.467**	**-1.32**	**54.88**
60	M	165	**57.667**	**0.233**	**0.21**	**58.11**
70	M	171	**57.667**	**1.483**	**1.335**	**60.485**
75	M	180	**57.667**	**1.483**	**1.335**	**60.485**

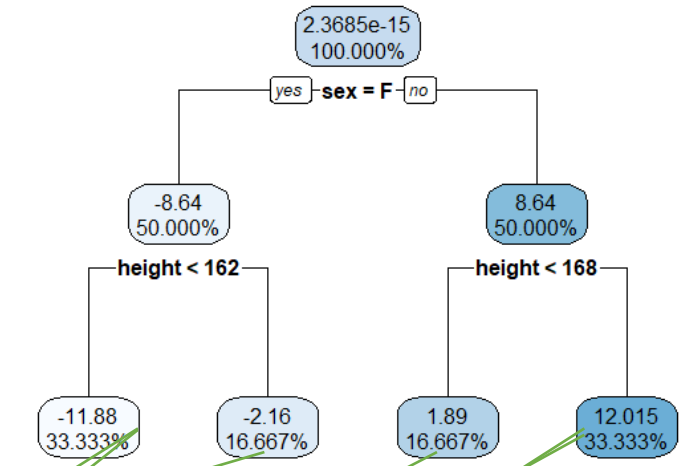
Gradient Boosting Regression (Step)

เราเริ่มทำต้นที่ 2 โดยการหา error ที่ model predicted ผลาด (error_2)
error_2 = weight - weight_2

weight	sex	height	weight_2	error_2
55	F	165	57.16	-2.16
45	F	159	54.88	-9.88
41	F	151	54.88	-13.88
60	M	165	58.11	1.89
70	M	171	60.485	9.515
75	M	180	60.485	14.515

weight	sex	height	weight_2	error_2	error_pred3
55	F	165	57.16	-2.16	-2.16
45	F	159	54.88	-9.88	-11.88
41	F	151	54.88	-13.88	-11.88
60	M	165	58.11	1.89	1.89
70	M	171	60.485	9.515	12.015
75	M	180	60.485	14.515	12.015

นำ error_2 ที่ได้มา ไปทำ Decision Tree model
โดย Y variable คือ error_2 , Feature คือ sex และ height



ต้นที่ 3

Gradient Boosting Regression (Step)

นำ error ที่ได้จาก decision tree (error_pred3) ไปคูณ การ learning rate (shrinkage rate/ alpha) assume ว่า learning rate = 0.1
error_pred3_alpha = error_pred3 * 0.1

weight	sex	height	weight_2	error_2	error_pred3	error_pred3_alpha
55	F	165	57.16	-2.16	-2.16	-0.216
45	F	159	54.88	-9.88	-11.88	-1.188
41	F	151	54.88	-13.88	-11.88	-1.188
60	M	165	58.11	1.89	1.89	0.189
70	M	171	60.485	9.515	12.015	1.2015
75	M	180	60.485	14.515	12.015	1.2015

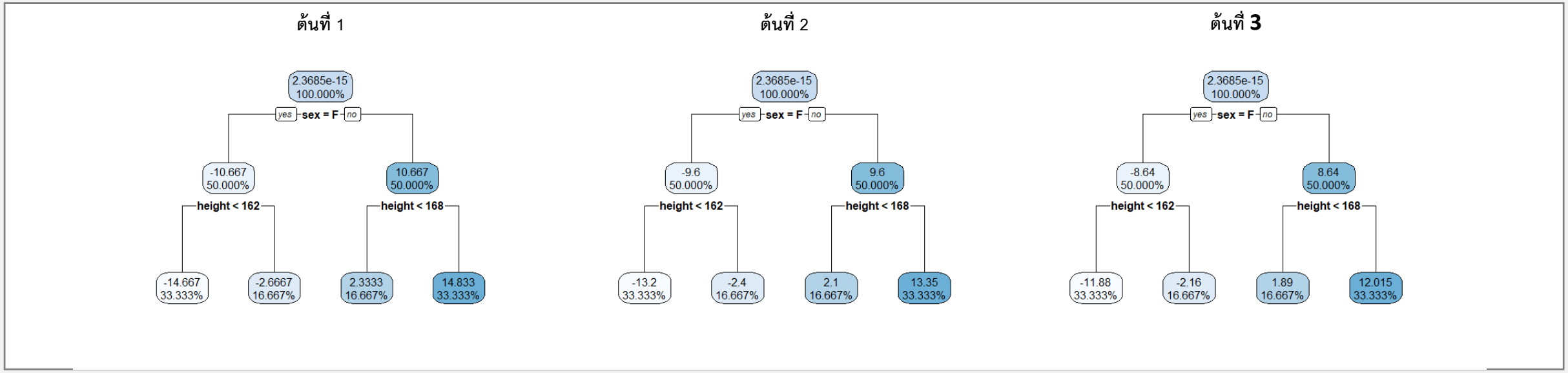
Weight_3 = average weight + error_pred1_alpha + error_pred2_alpha + error_pred3_alpha
(weight_3 เป็นค่า prediction จากต้นไม้ต้นที่ 3 final prediction)

weight	sex	height	average weight	error_pred1_alpha	error_pred2_alpha	error_pred3_alpha	weight_3
55	F	165	**57.667**	**-0.267**	**-0.24**	**-0.216**	**56.944**
45	F	159	**57.667**	**-1.467**	**-1.32**	**-1.188**	**53.692**
41	F	151	**57.667**	**-1.467**	**-1.32**	**-1.188**	**53.692**
60	M	165	**57.667**	**0.233**	**0.21**	**0.189**	**58.299**
70	M	171	**57.667**	**1.483**	**1.335**	**1.2015**	**61.6865**
75	M	180	**57.667**	**1.483**	**1.335**	**1.2015**	**61.6865**

Gradient Boosting Regression (Step)

weight	weight_3 (Final Prediction)
55	**56.944**
45	**53.692**
41	**53.692**
60	**58.299**
70	**61.6865**
75	**61.6865**

Gradient Boosting Regression (Step)



หากต้องการ prediction weight
โดยมี gender = M , height = 174

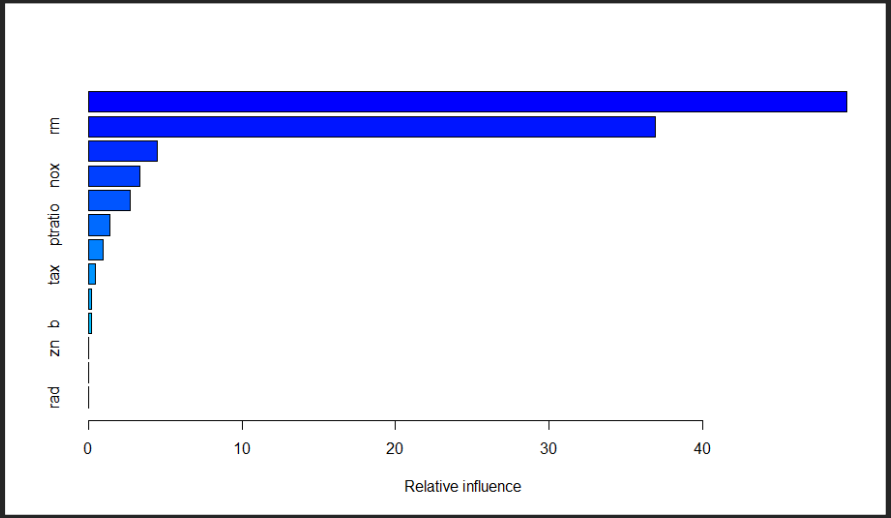
1. Average weight = 57.667
2. นำ gender , height ที่มีอยู่ไปใส่ ต้นไม้ทั้ง 3 เพื่อหาค่า predicted error (14.833 , 13.35 , 12.015)
3. นำ predicted error ไป scaled กับ learning rate *0.1 (1.4833 , 1.335 , 1.2015)
4. นำ ค่า error scaled ทุกค่าไปบวกกับ average weight ตอนแรก ก็จะได้ Final prediction result

   57.667 + 1.4833 +1.335+1.2015

   = **61.6868 KG**

Gradient Boosting Regression (R Code)

```
library(gbm)
set.seed(639)
gbm_model <- gbm(formula = medv~., distribution = "gaussian", data = BostonHousing,
                 n.trees = 20,interaction.depth = 5, n.minobsinnode = 10,
                 shrinkage = 0.1, bag.fraction = 1)
summary(gbm_model)

            var    rel.inf
lstat     lstat 49.4243616
rm           rm 36.9404910
dis         dis  4.4895090
nox         nox  3.3188951
crim       crim  2.7137412
ptratio ptratio  1.3515968
age         age  0.9723611
tax         tax  0.4146805
indus     indus  0.1913072
b             b  0.1830563
zn           zn  0.0000000
chas       chas  0.0000000
rad         rad  0.0000000
```



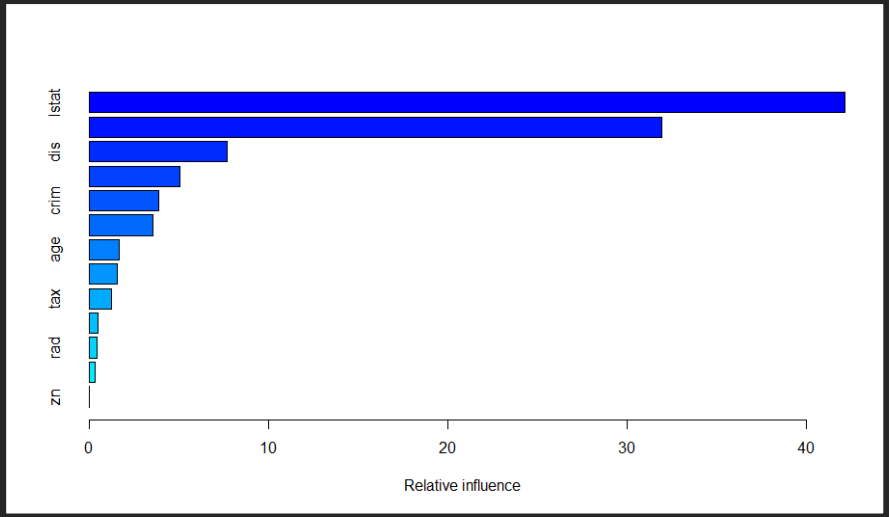
Gradient Boosting Regression (R Code Stochastic GBR)

```
library(gbm)
set.seed(639)
sgbm_model <- gbm(formula = medv~., distribution = "gaussian", data = BostonHousing,
                  n.trees = 20,interaction.depth = 5, n.minobsinnode = 10,
                  shrinkage = 0.1, bag.fraction = 0.8)
summary(sgbm_model)
```

Gradient Boosting Regression (R Code Caret Version)

Caret Version

```
library(caret)
library(gbm)

set.seed(639)
gbm_model_caret <- train(medv~., data = BostonHousing,  method = "gbm")

summary(gbm_model_caret)

            var    rel.inf
lstat     lstat 42.1650978
rm           rm 31.9380684
dis         dis  7.6836875
nox         nox  5.0723956
crim       crim  3.8598316
ptratio ptratio  3.5377629
age         age  1.6819143
b             b  1.5812837
tax         tax  1.2100527
chas1     chas1  0.4749773
rad         rad  0.4521296
indus     indus  0.3427985
zn           zn  0.0000000
```



Gradient Boosting Regression (Pros & Cons)

Pros

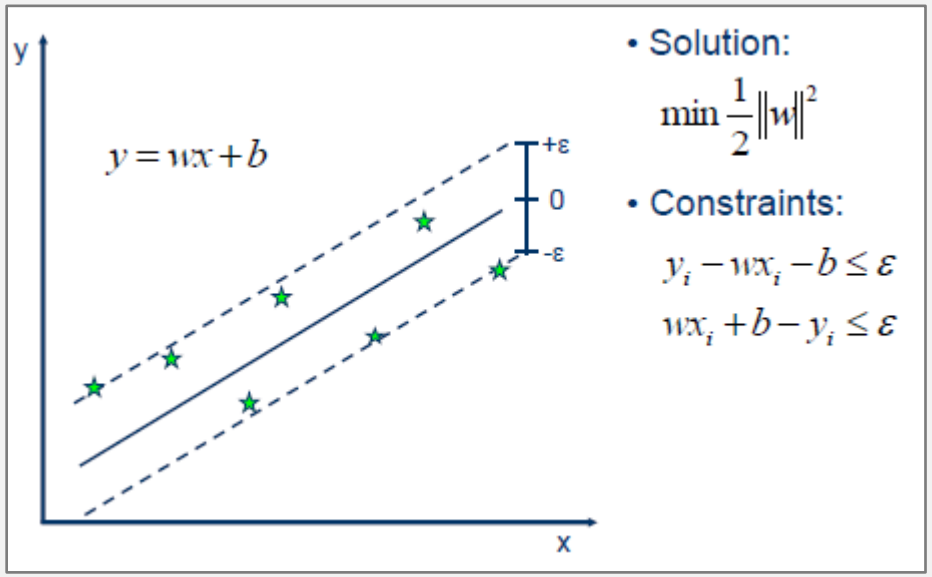
1. เป็น Model ที่ **Prediction แม่นมาก**
2. แม่นกว่า random forest หาก tune ดีๆ
3. ไม่ต้องทำการ preprocess feature

Cons

1. Model จะ **sensitive to Outliner** มาก
2. **ใช้เวลา Train มาก**
3. **Overfit ง่ายต้อง** tune ดีๆ (ควรจะให้ parameter learning rate (shrinkage rate) และ depth of tree น้อยๆ ไว้ เพื่อ ให้ algorithm เรียนแบบ ช้าๆ) .

Support Vector Machine Regression

Support Vector Regression (Linear Kernel)



ทำให้ ท่อ Flat ที่สุด

โดยให้ทุก Data Point อยู่ภายในท่อ

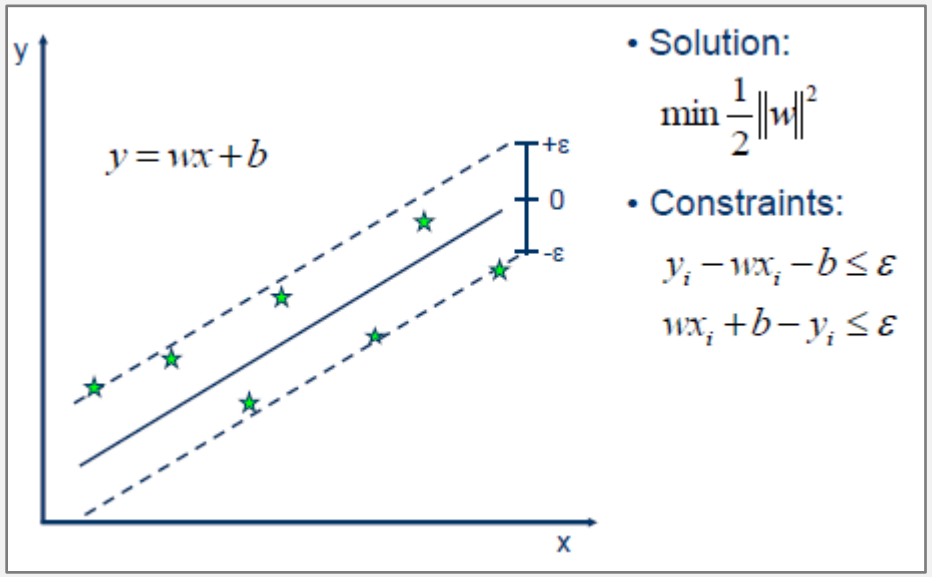
Note: svm เราจะต้อง min ||w|| ดังนั้นเราจำนวนต้องให้ w ของทุก feature ยุระดับเดียวกัน
ต้อง scale ข้อมูล

Objective : คือสร้าง **ท่อ** ที่ **Flat (นอนราบ)** มากที่สุด แล้วมีทุกๆ Data Point อยู่ในท่อ
โดย **ท่อ** จะประกอบด้วย
1. เส้นศูนย์กลาง (wx +b)
2. เส้นขอบบน (wx +b) + e
3. เส้นขอบล่าง (wx +b) - e

Ideally : ทุก Data Point ต้องอยู่ใน **ท่อ**

รูปจาก
http://www.saedsayad.com/support_vector_machine_reg.htm

Support Vector Regression (Linear Kernel)



สมการ เส้นตรง : y = ax +b

สมการเส้นศูนย์กลางของ SVR : y = wx +b

If | |w| |= 0 -> y = b (เส้นตรงที่ Flat มากที่สุด)

รูปจาก
http://www.saedsayad.com/support_vector_machine_reg.htm

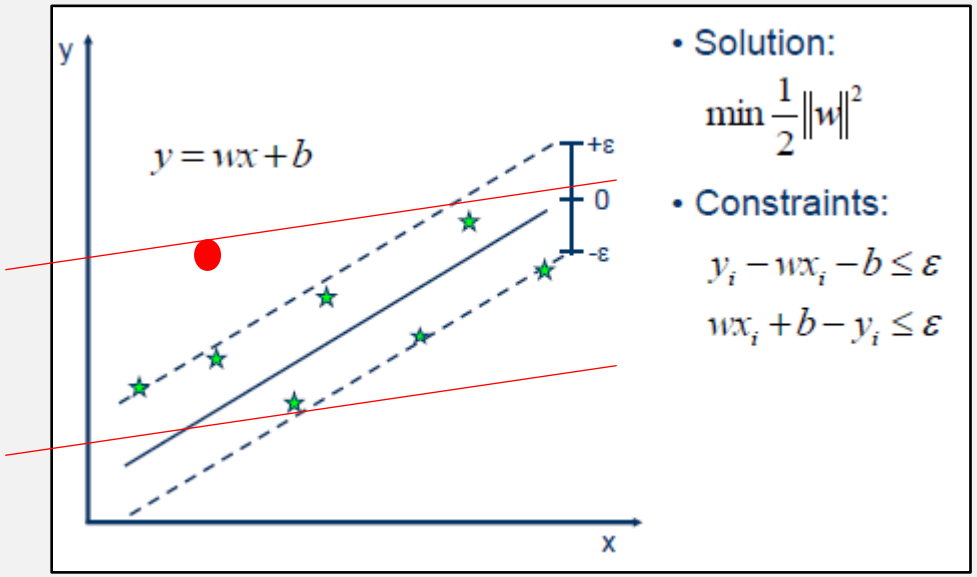
Objective : คือสร้าง **ท่อ** ที่ **Flat (นอนราบ)** มากที่สุด แล้วมีทุกๆ Data Point อยู่ในท่อ

โดย **ท่อ** จะประกอบด้วย

1. เส้นศูนย์กลาง (wx +b)
2. เส้นขอบบน (wx +b) + e
3. เส้นขอบล่าง (wx +b) - e

Ideally : ทุก Data Point ต้องอยู่ใน **ท่อ**

Support Vector Regression (Linear Kernel)



สมการ เส้นตรง : y = ax +b

สมการเส้นศูนย์กลางของ SVR : y = wx +b

If | |w| |= 0 -> y = b (เส้นตรงที่ Flat มากที่สุด)

รูปจาก
http://www.saedsayad.com/support_vector_machine_reg.htm

Objective : คือสร้าง **ท่อ** ที่ **Flat (นอนราบ)** มากที่สุด แล้วมีทุกๆ Data Point อยู่ในท่อ

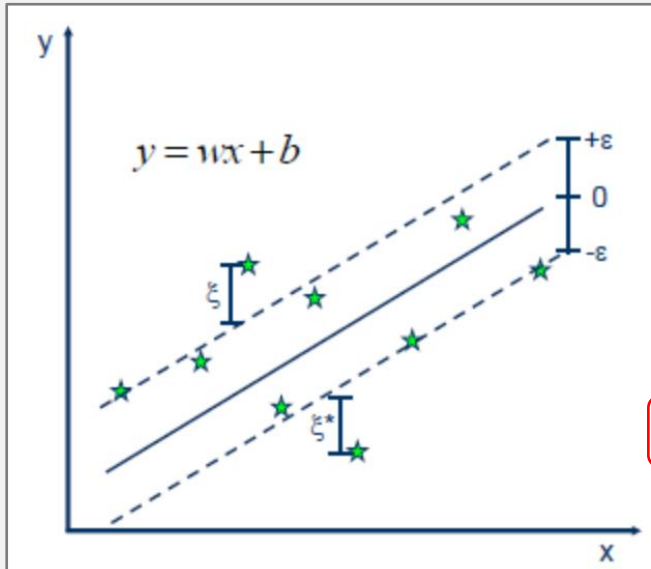
โดย **ท่อ** จะประกอบด้วย

1. เส้นศูนย์กลาง (wx +b)
2. เส้นขอบบน (wx +b) + e
3. เส้นขอบล่าง (wx +b) - e

Ideally : ทุก Data Point ต้องอยู่ใน **ท่อ**

แต่ความเป็นจริง : Data Point ให้ทุกตัวอยู่ในท่อ ยากมาก

Support Vector Regression (Linear Kernel : Soft Margin)



- Minimize:

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*)$$

- Constraints:

$$y_i - wx_i - b \le \varepsilon + \xi_i$$

$$wx_i + b - y_i \le \varepsilon + \xi_i^*$$

$$\xi_i, \xi_i^* \ge 0$$

Objective : คือสร้าง **ท่อ** ที่ **Flat (นอนราบ)** มากที่สุด โดยที่พยายามให้ Data Point หลุดออกจาก **ท่อ** จำนวนน้อย

ควบคุมด้วย Parameter C

C มาก -> เราให้น้ำหนัก Correct prediction มากกว่า **ท่อ** ควรจะ ครอบ Data Point ให้ได้มากที่สุด

C น้อย -> เราให้น้ำหนัก Flatness มากกว่า **ท่อ** ควรจะ ราบนอนให้ได้มากที่สุด

Practically : ทุก บาง Data point หลุดออกจากท่อได้

Support Vector Regression (Linear Kernel : Soft Margin)

Objective : คือสร้าง **ท่อ** ที่ **Flat (นอนราบ)** มากที่สุด โดยที่พยายามให้ Data Point หลุดออกจาก **ท่อ** จำนวนน้อย

ควบคุมด้วย Parameter C

C มาก -> เราให้น้ำหนัก Correct prediction มากกว่า **ท่อ** ควรจะ ครอบ Data Point ให้ได้มากที่สุด

C น้อย -> เราให้น้ำหนัก Flatness มากกว่า **ท่อ** ควรจะ ราบนอนให้ได้มากที่สุด

Tune for C

C มาก

ท่อ ครอบ Data Point มากเกินไป

Variance PROBLEM

C น้อย

ท่อ ไม่ครอบ Data Point เท่าที่ควร

BIAS PROBLEM

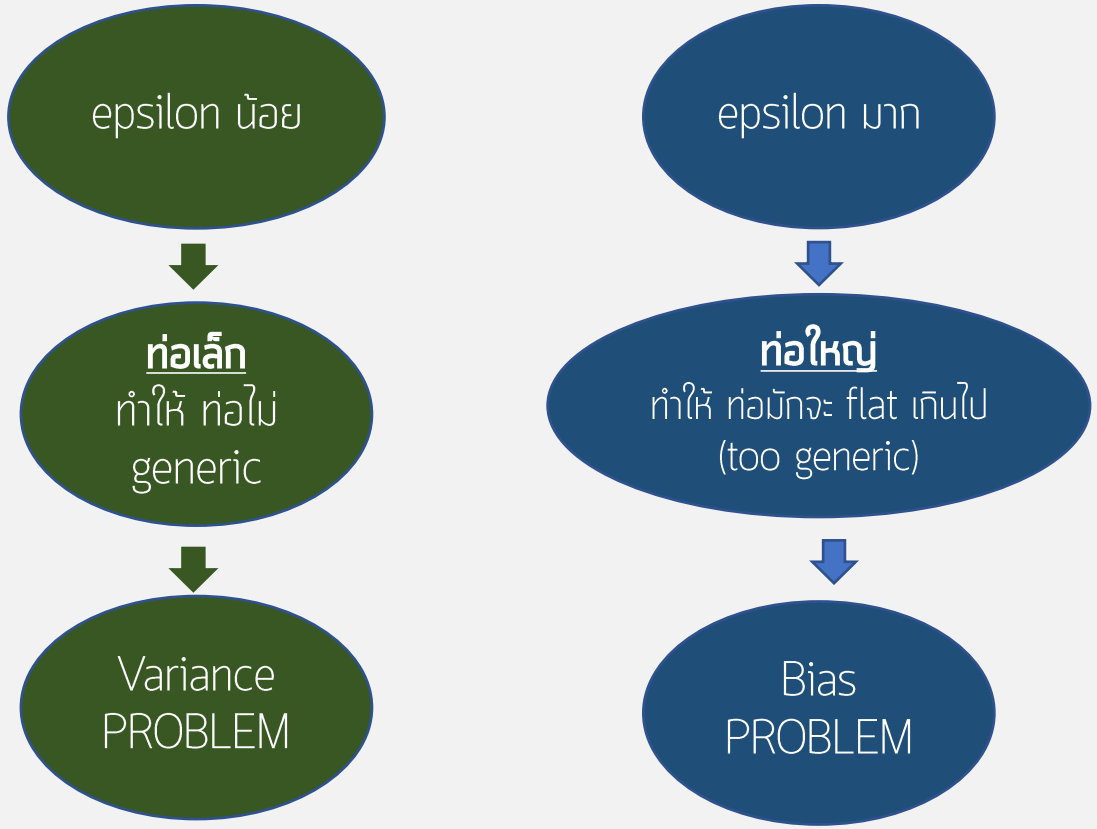
Support Vector Regression (Linear Kernel : Soft Margin)

ขนาดความกว้างของท่อ : Parameter ε (epsilon)

ε มาก -> ท่อ**กว้าง**มาก ทำให้**ท่อใหญ่**เกินไปและมักจะ flat เกินไป

ε น้อย -> ท่อ**เล็ก**มาก ทำให้**ท่อเล็ก**เกินไป ท่อนั้นพยายาม fit data มากเกินไป

(epsilon น้อยลง -> ท่อเล็กลง -> data หลุดออกจากท่อมากขึ้น -> algorithm พยายาม fit ท่อที่เล็กนั้นเข้ากับ data point ที่เพิ่งหลุดจากท่อต่อไป)



Support Vector Regression (Linear Kernel R Code)

[preprocess data]

```
library(mlbench)
library(e1071)
library(ggplot2)
library(caret)

## data

data("BostonHousing")
str(BostonHousing)

### split data
set.seed(4289)
num_train <- createDataPartition(BostonHousing$medv , p = 0.8 , list= FALSE)
training_data <- BostonHousing[num_train,]
testing_data <- BostonHousing[-num_train,]


training_feature <- training_data[ , -which(names(training_data)=="medv")]
scaling_model<- preProcess(training_feature , method = c("center","scale"))
training_scaled <- predict(scaling_model , training_data)
testing_scaled <- predict(scaling_model , testing_data)
```

Support Vector Regression (Linear Kernel R Code)

```
library(e1071)
linear_svm = svm(medv~. ,kernel = "linear", cost = 1 , epsilon = 0.1 ,data = training_scaled)
predict(linear_svm , newdata = testing_scaled)
```

Support Vector Regression (Linear Kernel R Code Caret)

Caret Version

```
library(kernlab)
library(caret)
set.seed(639)
linear_svm_caret <- train(medv ~.,
                          data = training_scaled,
                          method = 'svmLinear',
                          tuneGrid = expand.grid(C = 1))
predict(linear_svm_caret , newdata = testing_scaled)
```

SVM Regression (Pros & Cons)

Pros

1. หาก ข้อมูลเรา feature เยอะเมื่อเทียบกับ จำนวน datapoint svm จะ perform ดีกว่า model อื่นๆ
2. จัดการ outlier ได้ดี

Cons

1. Train ช้ามาก (โดยเฉพาะ dataset ใหญ่ๆ)
2. **Overfit ง่าย**

Support Vector Regression (Kernel Trick [Poly , RBF])

Support Vector Regression (Kernel Trick)

สมมุติเรามีแค่ 2-dimension X , Y

X,Y

เส้นตรง fit ได้ไม่ดี ใน 2 dimension
(2 แกน : X, Y)

เปิด **dimension** ใช้ **kernel**

Z1 , Z2 , Z3 , Z4 , Z5

ใช้ plane หรือ เส้นตรงในรูปแบบหลาย dimension
เพื่อ fit ข้อมูลในหลาย dimension
(dimension มี Z1 , Z2 , Z3 , Z4 , Z5)
Fit ข้อมูลได้ดีขึ้น

solution

plane หรือ เส้นตรงที่ดีที่สุด ในหลาย dimension

Transform กลับมา เป็น 2 **dimension**

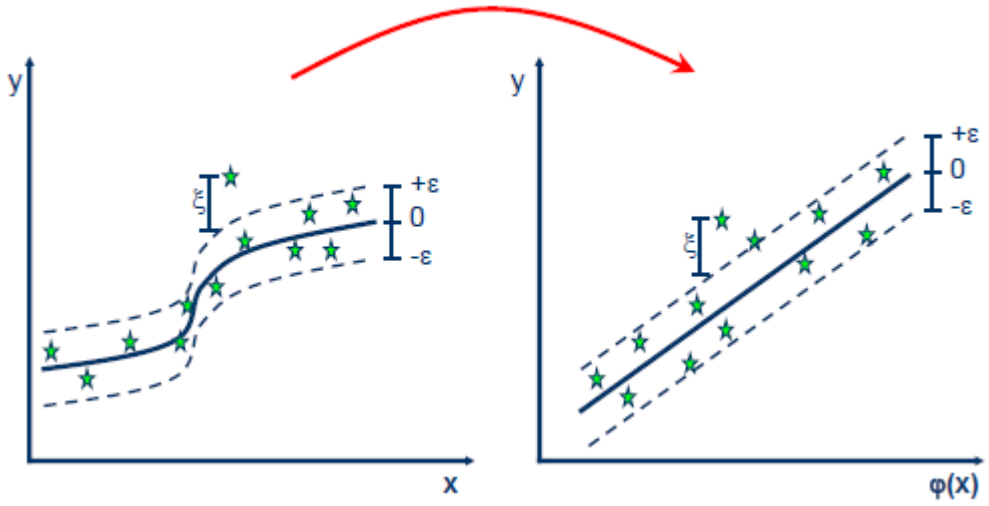
เส้น nonlinear ใน 2 **dimension**

เส้น nonlinear ที่ fit ข้อมูลได้ดีขึ้น

Note : ความหมายของ **dimension** คือ **feature** ใหม่ หรือ **column feature** ใหม่
ที่ใช้ **Term dimension** เพราะว่าเวลา **plot graph** มันจะมี **1 feature** ต่อ **1 dimension**
เช่น ถ้า **plot graph y , x** จะมี **2 dimension**
ถ้า **plot graph y , x1 , x2**

Support Vector Regression (Kernel Trick)

Data Transformation



ไส้ในของ สมการ SVR เราสามารถ พลิก สูตรเป็น แบบนี้ได้

$$y = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) \cdot \langle \varphi(x_i), \varphi(x) \rangle + b$$

เราจะสามารถ save time + resource โดยการใช้ สูตรนี้แทนได้

$$y = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) \cdot K(x_i, x) + b$$

แต่ใน SVR เราไม่ต้อง compute dot product ระหว่าง datapoint เราแค่ใช้ Kernel function ในการ Transform data ของเรา (เช่น Polynomial Kernel)

Support Vector Regression (Kernel Trick)

<u>โดย Kernel function ที่ใช้บ่อยๆจะมี</u>

1. Linear kernel
2. Polynomial Kernel
3. Radial Basis function kernel

Polynomial kernel

$$k(\mathbf{x}_1, \mathbf{x}_2) = (\gamma\, \mathbf{x}_1 \cdot \mathbf{x}_2 + c)^d$$

RBF kernel

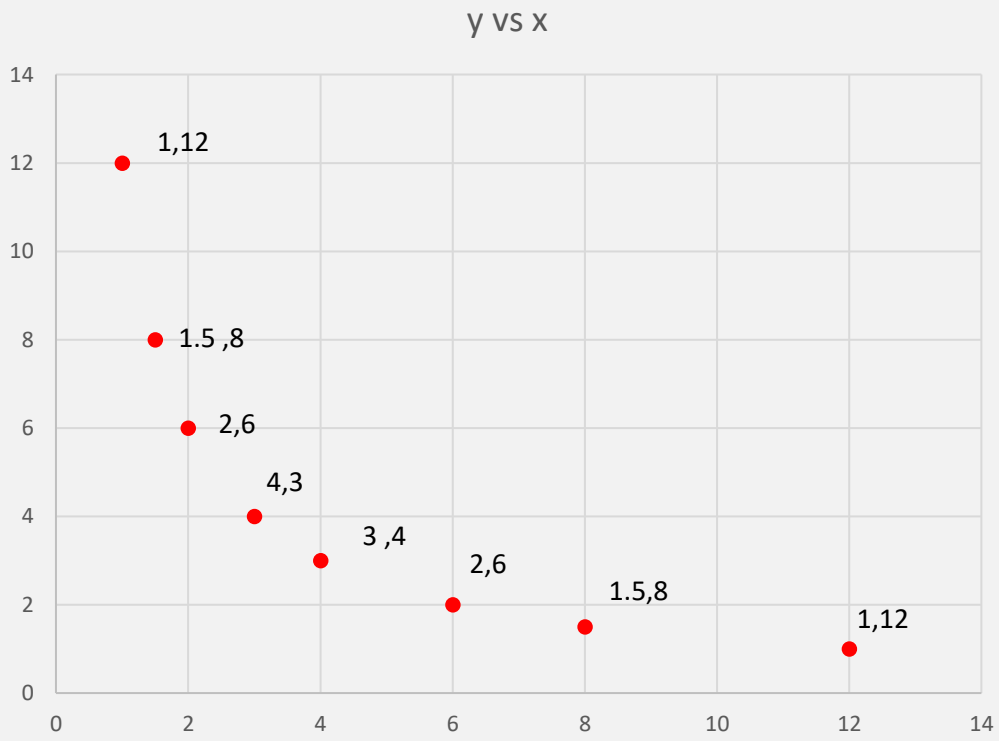
$$k(\mathbf{x}_1, \mathbf{x}_2) = \exp\left(-\gamma \left\| \mathbf{x}_1 - \mathbf{x}_2 \right\|^2\right)$$

$$y = \sum_{i=1}^{N} \left(\alpha_i - \alpha_i^*\right) \cdot K(x_i, x) + b$$

Polynomial kernel

- เปิด dimension โดยการ เพิ่ม degree ของ polynomial ลักษณะจะคล้ายๆ polynomial regression
- สมมุติว่าเรามี dataset ตามข้างล่าง ที่ไม่สามารถ fit เส้นตรงกับ ข้อมูลใน 2-dimension ได้

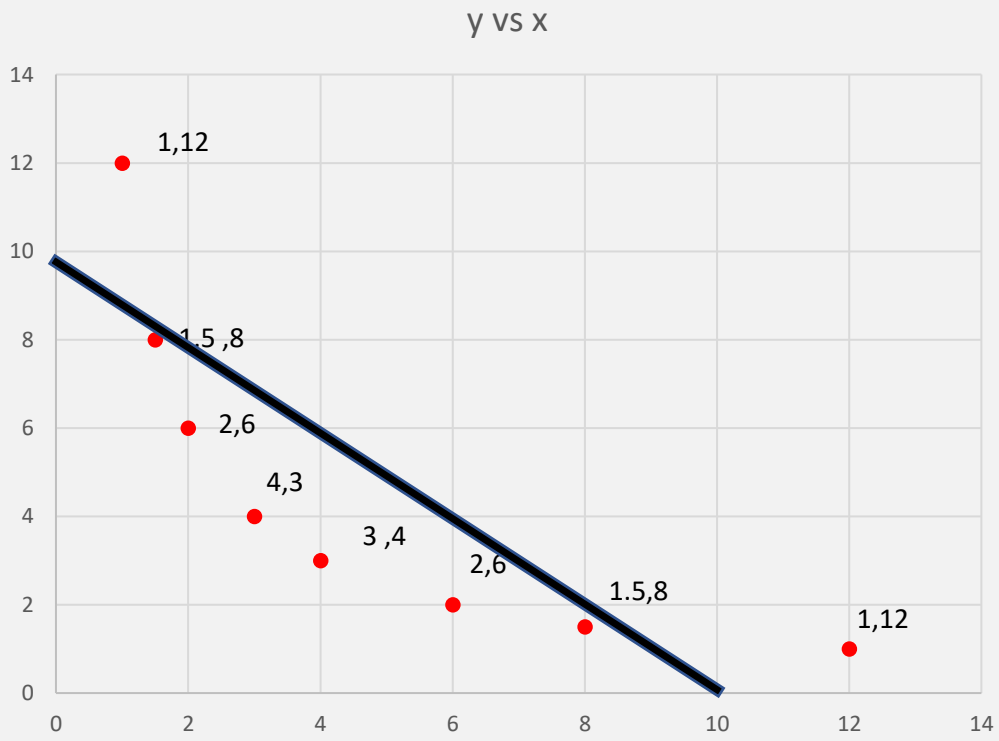
y	x
12	1
8	1.5
6	2
4	3
3	4
2	6
1.5	8
1	12



Polynomial kernel

- เปิด dimension โดยการ เพิ่ม degree ของ polynomial ลักษณะจะคล้ายๆ polynomial regression
- สมมุติว่าเรามี dataset ตามข้างล่าง ที่ไม่สามารถ fit เส้นตรงกับ ข้อมูลใน 2-dimension ได้

y	x
12	1
8	1.5
6	2
4	3
3	4
2	6
1.5	8
1	12



Polynomial kernel

- เพิ่ม dimension เข้าไป
  - dimension ไหนที่ fit ข้อมูลได้ดี ? **(X^2 , Y^2 , X*Y)**

y	**x**	Y^2	X^2	X*Y
12	1	144	1	12
8	1.5	64	2.25	12
6	2	36	4	12
4	3	16	9	12
3	4	9	16	12
2	6	4	36	12
1.5	8	2.25	64	12
1	12	1	144	12

Polynomial kernel

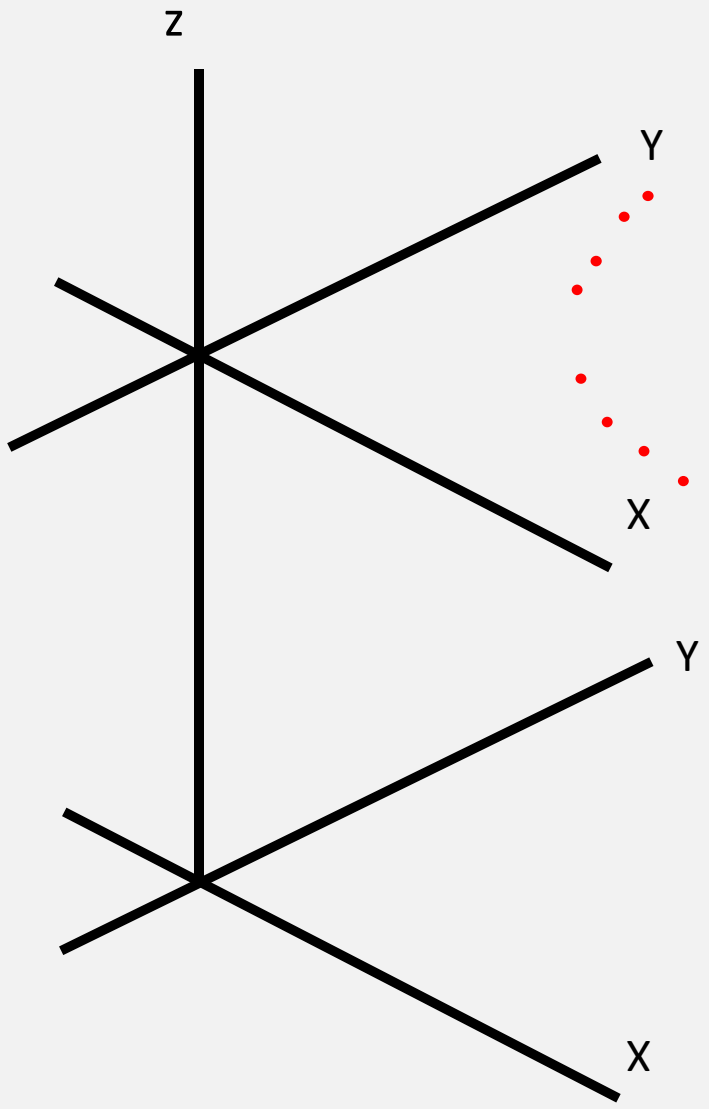
- เพิ่ม dimension เข้าไป
  - dimension ไหนที่ fit ข้อมูลได้ดี ? **(X^2 , Y^2 , X*Y)**

y	**x**	Y^2	X^2	**X*Y**
12	1	144	1	**12**
8	1.5	64	2.25	**12**
6	2	36	4	**12**
4	3	16	9	**12**
3	4	9	16	**12**
2	6	4	36	**12**
1.5	8	2.25	64	**12**
1	12	1	144	**12**

Support Vector Regression (Polynomial kernel)

เปลี่ยน data 3 column ให้เป็น graph

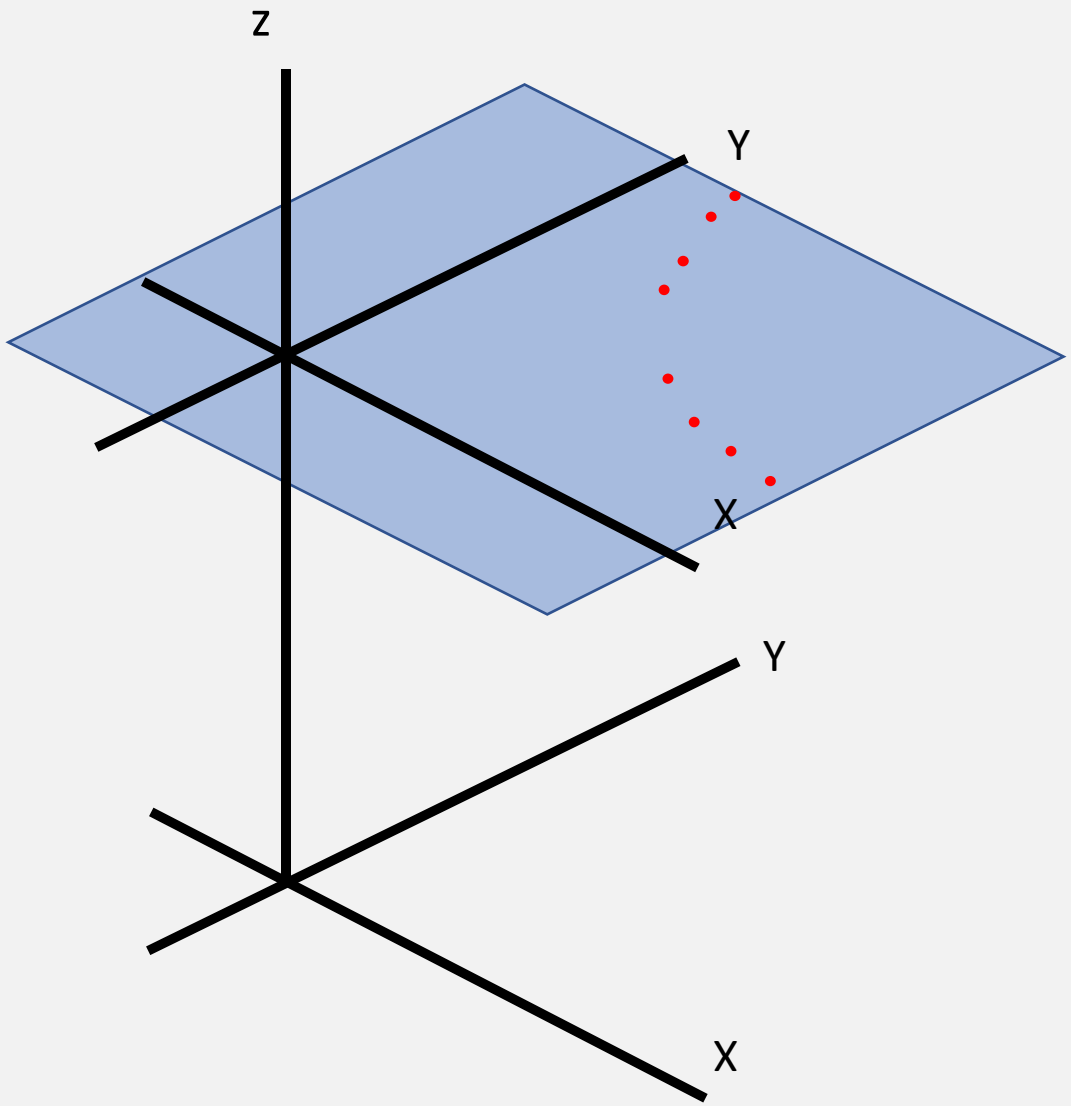
y	x	Z = X*Y
12	1	**12**
8	1.5	**12**
6	2	**12**
4	3	**12**
3	4	**12**
2	6	**12**
1.5	8	**12**
1	12	**12**



Support Vector Regression (Polynomial kernel)

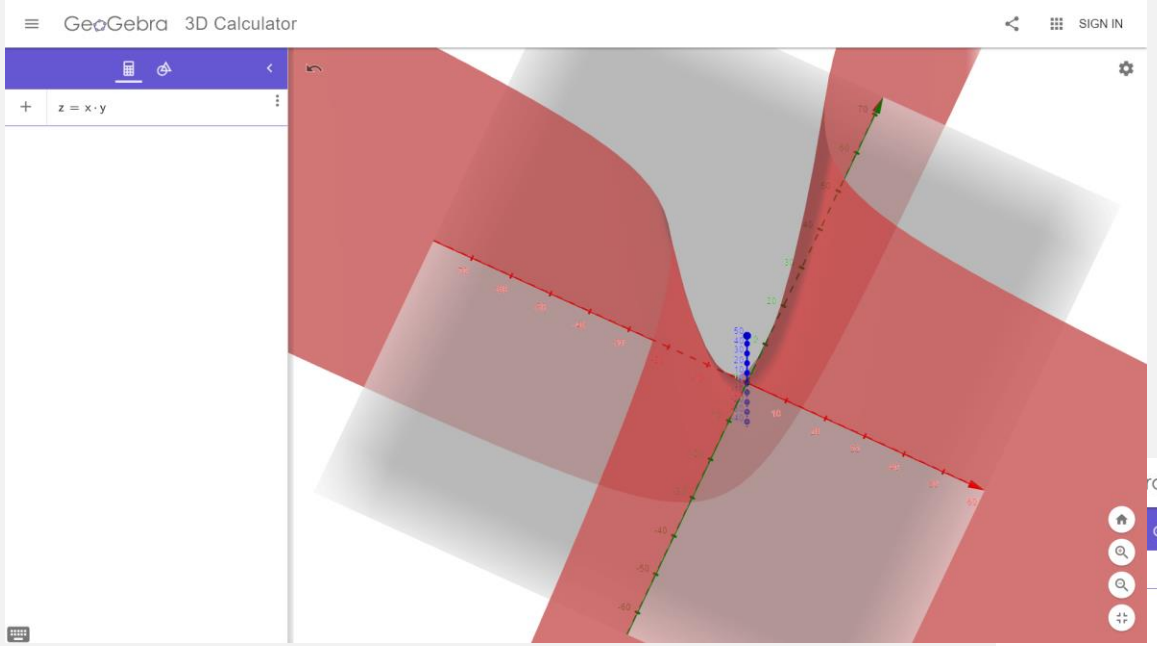
หากเป็น 3d ใช้ plane fit ข้อมูลได้ดีที่สุด

y	x	Z = X*Y
12	1	**12**
8	1.5	**12**
6	2	**12**
4	3	**12**
3	4	**12**
2	6	**12**
1.5	8	**12**
1	12	**12**



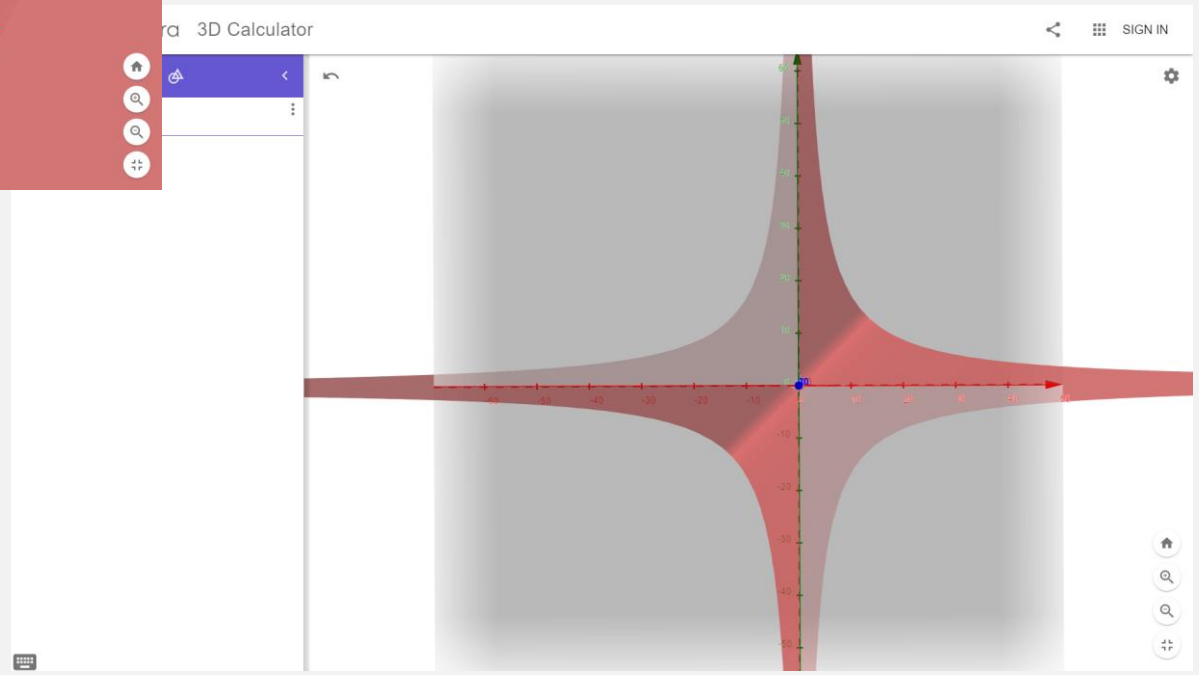
Support Vector Regression (Polynomial kernel)

Transform กลับมาเป็น 2 dimension



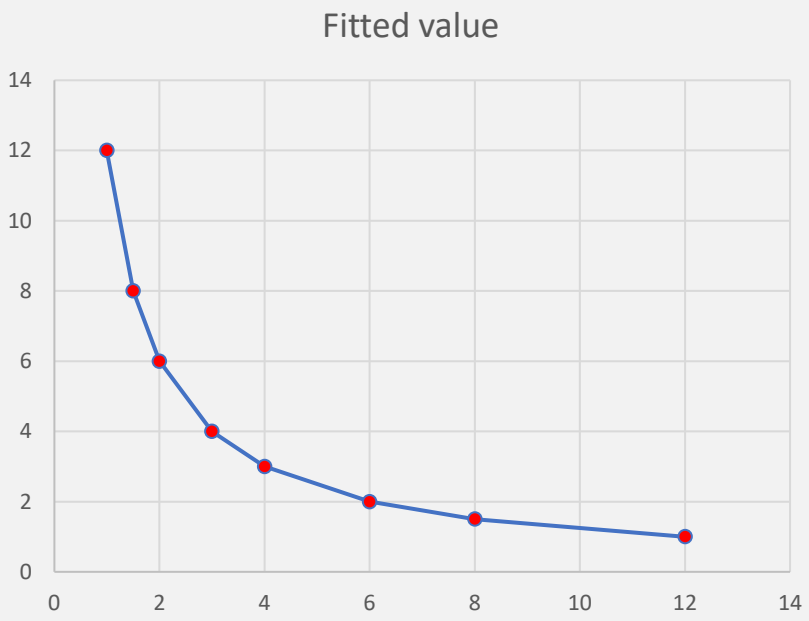
https://www.geogebra.org/3d?lang=en

Equation : z = x * y

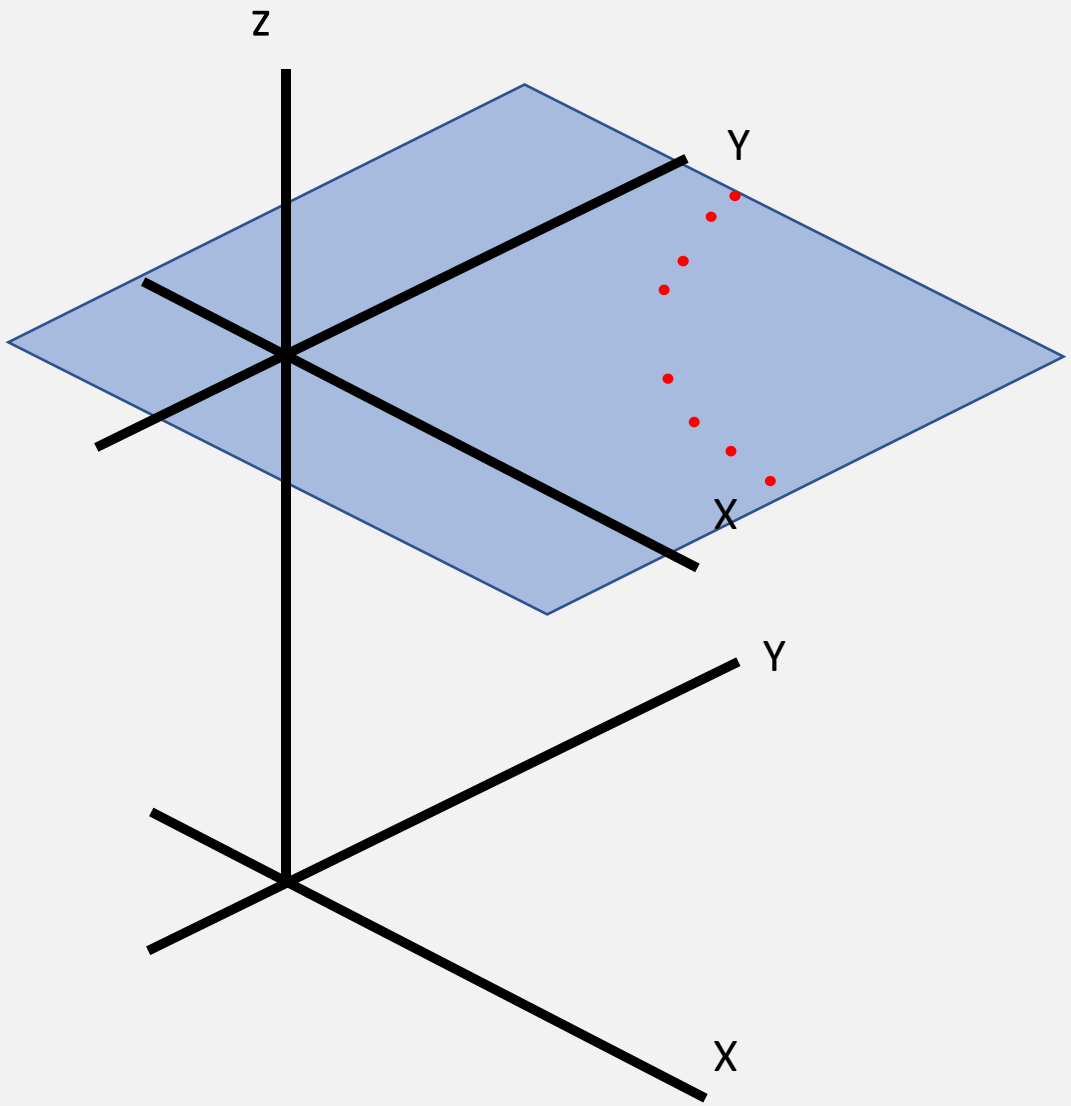


Support Vector Regression (Polynomial kernel)

Transform กลับมาเป็น 2 dimension



Z = Y* X



Dimension vs Degree of Polynomial

- ตัวอย่าง dimension ที่ถูกเปิดขึ้นมา กับ hyperparameter degree of polynomial
  - d = 2
    - เราจะเปิดได้ 5 dimension : **Y , X , X^2 , X*Y , Y^2**
  - d = 3
    - เราจะเปิดได้ 9 dimension : **Y , X , X^2 , X*Y , Y^2 , X^3 , (X^2)*Y , X*(Y^2) , Y^3**

Formula and hyperparameter

$$k(\mathbf{x}_1, \mathbf{x}_2) = (\gamma\, \mathbf{x}_1 \cdot \mathbf{x}_2 + c)^d$$

Hyper parameter ที่เราจะต้อง set คือ

Scale / gramma : ตัวที่ scale relationship ระหว่าง datapoint

Degree : degree of polynomial จะกำหนด dimension ที่เราจะเปิด

C : (ไม่ใช้ c ข้างบน) คือ ตัวกำหนด objective ระหว่าง correct prediction vs flatness

Epsilon (ไม่มีในการ tuning ของ caret) : ความกว้างของท่อ

Offset (ไม่มีในการ tuning ของ caret) : ตัว c ในสมการข้างบน

Formula and hyperparameter

$$k(\mathbf{x}_1, \mathbf{x}_2) = (\gamma\, \mathbf{x}_1 \cdot \mathbf{x}_2 + c)^d$$

Hyper parameter ที่เราจะต้อง set คือ

Scale / gramma : ตัวที่ scale relationship ระหว่าง datapoint

Degree : degree of polynomial จะกำหนด dimension ที่เราจะเปิด

C : (ไม่ใช้ c ข้างบน) คือ ตัวกำหนด objective ระหว่าง correct prediction vs flatness

Epsilon (ไม่มีในการ tuning ของ caret) : ความกว้างของท่อ

Offset (ไม่มีในการ tuning ของ caret) : ตัว c ในสมการข้างบน

Support Vector Regression (Polynomial kernel R code)

```
library(e1071)
poly_svm = svm(medv~. ,kernel = "poly",  degree = 4,  cost = 1 , epsilon = 0.1 ,data = training_scaled)
summary(poly_svm)

Call:
svm(formula = medv ~ ., data = training_scaled, kernel = "poly", degree = 4, cost = 1, epsilon = 0.1)


Parameters:
   SVM-Type:  eps-regression
 SVM-Kernel:  polynomial
       cost:  1
     degree:  4
      gamma:  0.07142857
     coef.0:  0
    epsilon:  0.1


Number of Support Vectors:  305




predict(poly_svm , newdata = testing_scaled)
```

Support Vector Regression (Polynomial kernel R Code Caret)

Caret Version

```
library(kernlab)
library(caret)
set.seed(639)
poly_svm_caret <- train(medv ~.,
                        data = training_scaled,
                        method = 'svmPoly',
                        tuneGrid = expand.grid(degree = 3,
                                               scale = 1,
                                               C = 1))
poly_svm_caret
Support Vector Machines with Polynomial Kernel

407 samples
 13 predictor

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 407, 407, 407, 407, 407, 407, ...
Resampling results:

  RMSE      Rsquared   MAE
  15.21395  0.2815242  6.244923

Tuning parameter 'degree' was held constant at a value of 3
Tuning parameter 'scale' was held constant at a value
 of 1
Tuning parameter 'C' was held constant at a value of 1
predict(poly_svm_caret , newdata = testing_scaled )
```

Intro

- Polynomial kernel เปิด dimension ตาม degree ที่เราใส่ เข้าไป
- RBF kernel เปิด **infinity dimension !!!**

formula

$$k(\mathbf{x}_1, \mathbf{x}_2) = \exp\left(-\boxed{\gamma}\left\|\mathbf{x}_1 - \mathbf{x}_2\right\|^2\right)$$

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2}{2\boxed{\sigma}^2}\right)$$

Hyper parameter ที่เราจะต้อง set คือ

gramma หรือ **sigma** : ตัวที่ scale relationship ระหว่าง datapoint

gramma เยอะ หรือ sigma น้อย : ข้อมูลที่ใกล้ท่อ มี influence ต่อการ prediction มากกว่าตัวไกลท่อ **มาก (overfit)**

gramma น้อย หรือ sigma เยอะ :ข้อมูลที่ใกล้ท่อ มี influence ต่อการ prediction มากกว่าตัวไกลท่อ **ไม่มาก (underfit)**

C : คือ ตัวกำหนด objective ระหว่าง correct prediction vs flatness

Epsilon (ไม่มีในการ tuning ของ caret) : ความกว้างของท่อ

Support Vector Regression (RBF kernel R Code)

```
library(e1071)
radial_svm = svm(medv~.
                  ,kernel = 'radial'
                  ,gamma = 1/ncol(training_scaled)
                  ,cost = 1
                  ,epsilon = 0.1
                  ,data = training_scaled)
radial_svm

Call:
svm(formula = medv ~ ., data = training_scaled, kernel = "radial", gamma =
1/ncol(training_scaled),
    cost = 1, epsilon = 0.1)


Parameters:
   SVM-Type:  eps-regression
 SVM-Kernel:  radial
       cost:  1
      gamma:  0.07142857
    epsilon:  0.1


Number of Support Vectors:  278

predict(radial_svm , newdata = testing_scaled )
```

Support Vector Regression (RBF kernel R Code Caret)

Caret Version

```
library(kernlab)
library(caret)
set.seed(639)
RBF_svm_caret <- train(medv ~.,
                       data = training_scaled,
                       method = 'svmRadial',
                       tuneGrid = expand.grid(
                         sigma = 0.1,
                         C = 1))
RBF_svm_caret
Support Vector Machines with Radial Basis Function Kernel

407 samples
 13 predictor

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 407, 407, 407, 407, 407, 407, ...
Resampling results:

  RMSE      Rsquared   MAE
  4.097471  0.8026793  2.543877

Tuning parameter 'sigma' was held constant at a value of 0.1
Tuning parameter 'C' was held constant at a value of 1
predict(RBF_svm_caret , newdata = testing_scaled )
```

SVM Regression kernel trick (Pros & Cons)

Pros

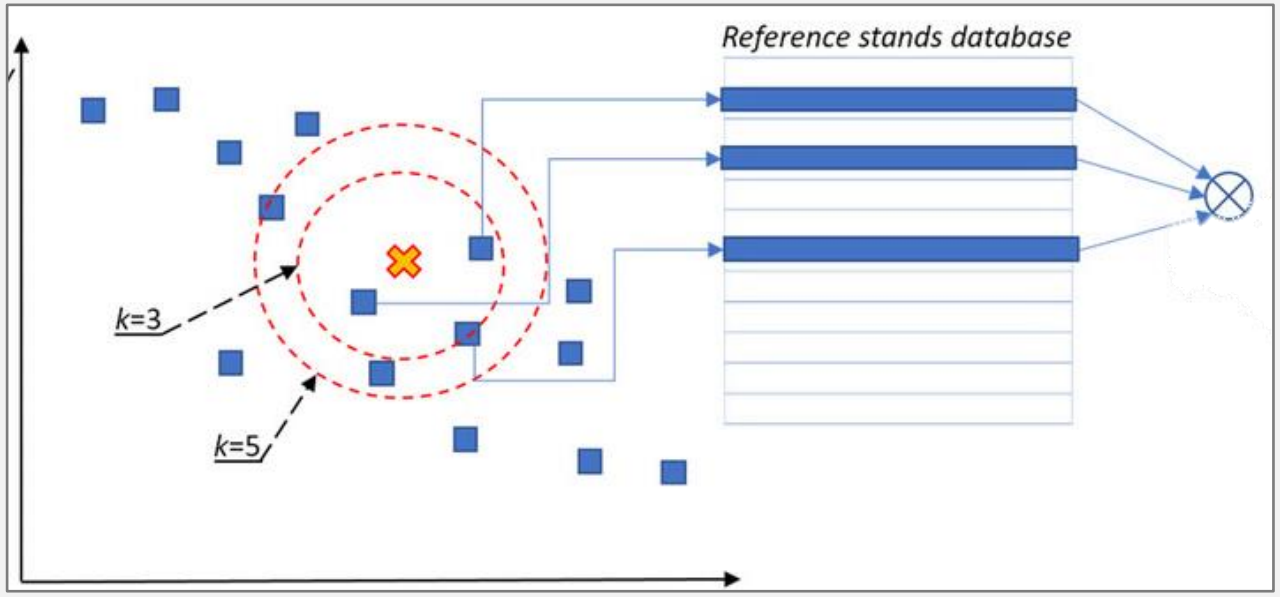
1. หาก ข้อมูลเรา feature เยอะเมื่อเทียบกับ จำนวน datapoint svm จะ perform ดีกว่า model อื่นๆ
2. จัดการ outlier ได้ดี
3. พวก polynomial , rbf kernel ค่อนข้างให้ prediction ที่แม่นเมื่อเทียบกับ model อื่น

Cons

1. Train ช้ามาก (โดยเฉพาะ dataset ใหญ่ๆ)
2. **Overfit ง่าย**

Distance Regression

Distance Regression (K-Nearest Neighbor Regression)

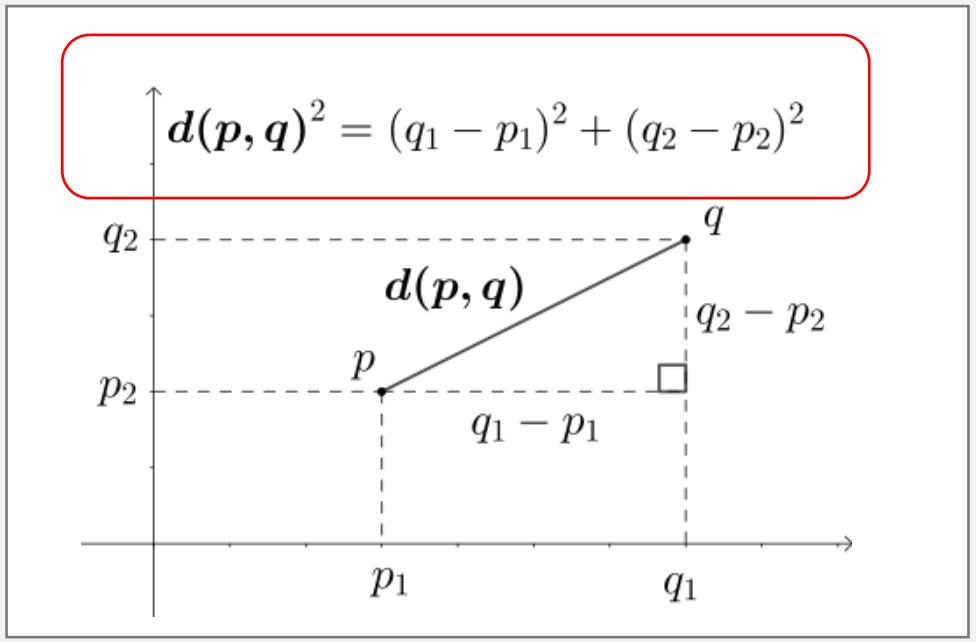


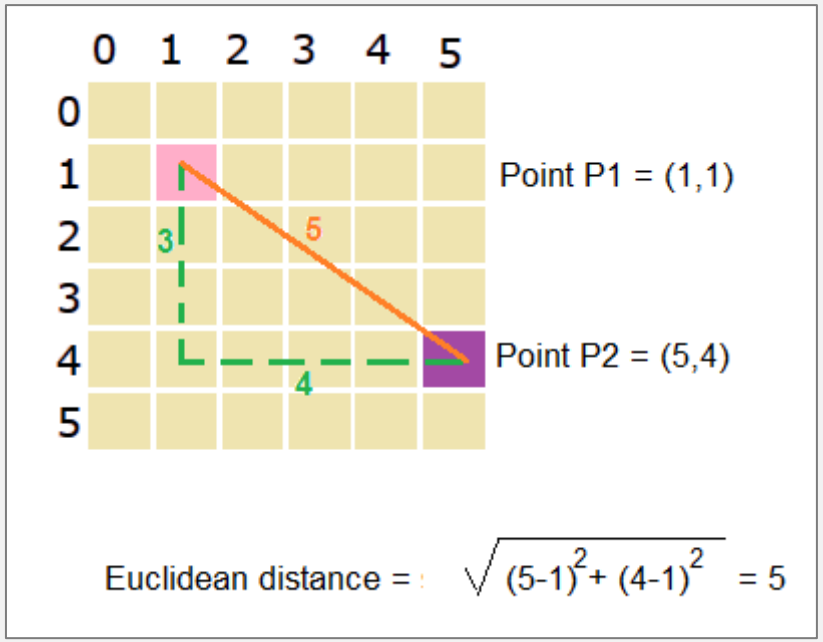
ระบุ K

(ex, K= 3) แล้วหา Data Point ที่ใกล้ เคียงจุด reference มากที่สุด **3** จุดแล้วนำมาหา Average ของ **3** จุดนั้น

(ex, K= 5) แล้วหา Data Point ที่ใกล้ เคียงจุด reference มากที่สุด **5** จุดแล้วนำมาหา Average ของ **5** จุดนั้น

การหา Distance ระหว่างจุด **Euclidean distance** :





เนื่องจาก เป็น **Algorithm Distance Based**: จำเป็นต้องให้ หน่วย feature ใกล้เคียงกัน -> **Scaling!!!!**
เช่น Minmax Scaler , Standard Scaler

Distance Regression (KNN R Code)

```
library(caret)
num_train <- createDataPartition(y= BostonHousing$medv , p =0.70 , list = FALSE)
train_data <- BostonHousing[num_train,]
test_data <- BostonHousing[-num_train,]
train_feature <- train_data[, -which(names(train_data) == "medv")]
scaling_model <- preProcess(train_feature, method = c("center","scale"))
train_scaled <- predict(scaling_model , newdata =train_data)
test_scaled <- predict(scaling_model , newdata =test_data)


knn_model <- knnreg(formula = medv ~., data = train_scaled , k = 5)
summary(knn_model)
```

Distance Regression (KNN R Code Caret)

Caret Version

```
library(caret)
scale_param <- preProcess(BostonHousing , method = c("scale", "center"))
BostonHousing_transform <- predict(scale_param, BostonHousing)
knn_model_caret <- train(medv ~. ,
                            data = BostonHousing_transform ,
                            method = "knn" ,
                            tuneGrid = expand.grid(
                             k = 5)
                            )
knn_model_caret

k-Nearest Neighbors

506 samples
 13 predictor

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 506, 506, 506, 506, 506, 506, ...
Resampling results:

  RMSE       Rsquared   MAE
  0.4467091  0.8091741  0.2982011

Tuning parameter 'k' was held constant at a value of 5
```

Distance Regression (KNN Regression Pros & Cons)

ข้อดี :

1. **ไม่ต้องมี Assumption เกี่ยวกับ Data** เช่น Linear relationship
2. เข้าใจง่าย และ อธิบายง่าย

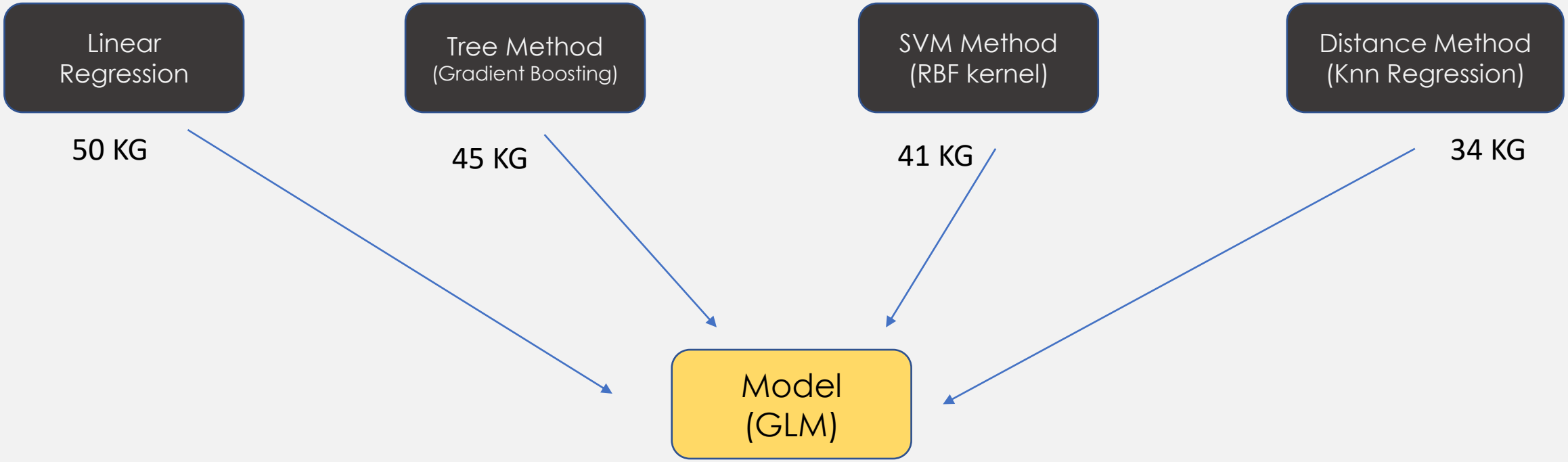
ข้อเสีย :

1. **ใช้ computing Resource เยอะมาก** ยิ่ง Dataset มี Row/Column เยอะยิ่งใช้เยอะมากๆ และใช้ Ram เยอะ
2. Sensitive กับ feature ที่ไม่มีพลัง
3. ต้อง **scale** ข้อมูลก่อน
4. หาก **feature เยอะ** : model จะ perform แย่
5. **Outlier** ทำร้าย model นี้

Ensemble Regression

Ensemble Regression

Prediction ของเราจะ**ไม่เพิ่ง result จาก model เดียว** แต่จะ depend จาก หลาย model
หลักการเหมือนกะ Random Forest ทำ **Model หลายรูปแบบ แล้วนำค่า prediction แต่ละ model มา เข้า model ซ้ำไป อีกที เช่น glm เพื่อไป predict actual value ต่อ**:



Glm model on (Actual ~ pred1 + pred2 + pred3 + pred4) = FINAL VALUE

Ensemble Regression (R Code (caretEnsemble))

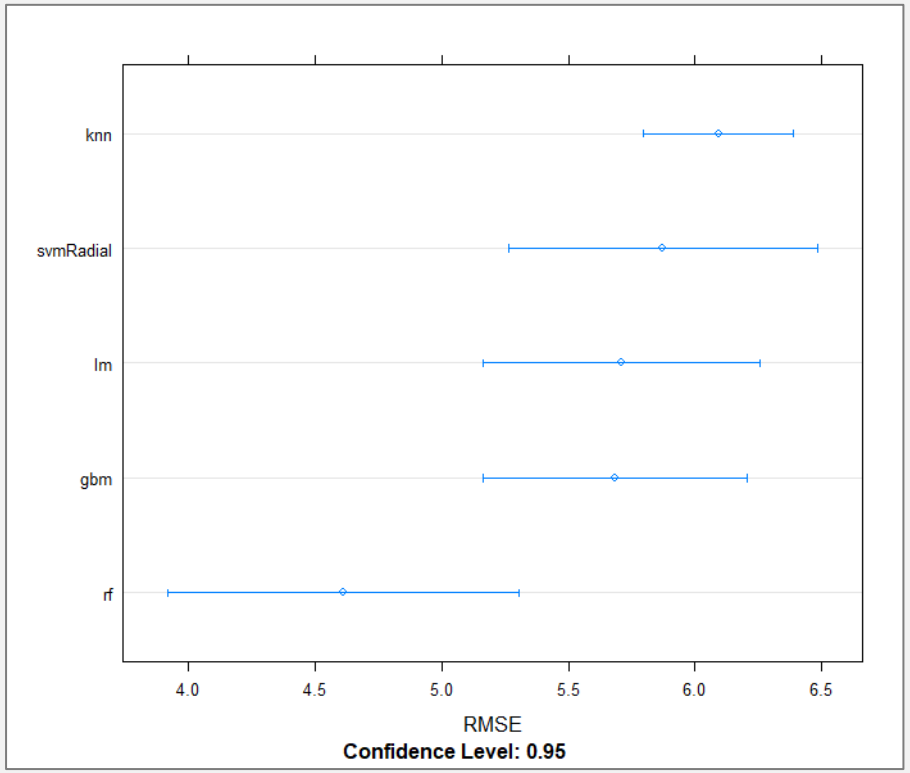
```
library(caretEnsemble)
library(caret)
set.seed(168)
df <- BostonHousing
num_train <- createDataPartition(y= df$medv , p =0.7 , list = FALSE)
train_data <- df[num_train,]
test_data <- df[-num_train,]
tc <- trainControl(method = "cv" , number = 5 ,index = createFolds(train_data$medv, 5) , savePredictions = "final")
model_list <- caretList(medv~. , data =  train_data ,trControl = tc
                                        , methodList = c("lm","svmRadial","rf","gbm","knn")
                                        ,continue_on_fail = FALSE , preProces = c("center","scale") )
```

```
model_list
$lm
Linear Regression

356 samples
 13 predictor

Pre-processing: centered (13), scaled (13)
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 72, 70, 71, 72, 71
Resampling results:

  RMSE     Rsquared   MAE
  5.70808  0.6270635  3.902222

Tuning parameter 'intercept' was held constant at a value of TRUE

$svmRadial
Support Vector Machines with Radial Basis Function Kernel

356 samples
 13 predictor

Pre-processing: centered (13), scaled (13)
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 72, 70, 71, 72, 71
Resampling results across tuning parameters:

  C     RMSE      Rsquared   MAE
  0.25  6.958194  0.5223351  4.208026
  0.50  6.310220  0.5772312  3.717975
  1.00  5.873460  0.6166559  3.432509

Tuning parameter 'sigma' was held constant at a value of 0.1019917
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were sigma = 0.1019917 and C = 1.

$rf
Random Forest

356 samples
 13 predictor

Pre-processing: centered (13), scaled (13)
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 72, 70, 71, 72, 71
Resampling results across tuning parameters:

  mtry  RMSE      Rsquared   MAE
   2    5.431791  0.6809234  3.387058
   7    4.716307  0.7518316  3.040035
  13    4.611853  0.7569402  3.008297

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was mtry = 13.
```

```
$gbm
Stochastic Gradient Boosting

356 samples
 13 predictor

Pre-processing: centered (13), scaled (13)
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 72, 70, 71, 72, 71
Resampling results across tuning parameters:

  interaction.depth  n.trees  RMSE      Rsquared   MAE
  1                   50      5.927784  0.5848729  4.060411
  1                  100      5.824683  0.5944818  4.012831
  1                  150      5.901826  0.5868578  4.107578
  2                   50      5.710104  0.6162293  3.870423
  2                  100      5.684035  0.6146612  3.893889
  2                  150      5.746937  0.6064597  4.015795
  3                   50      5.844474  0.5948583  3.973106
  3                  100      5.773631  0.6021678  3.941639
  3                  150      5.820243  0.5943924  4.016921

Tuning parameter 'shrinkage' was held constant at a value of 0.1
Tuning parameter 'n.minobsinnode' was held constant at a value of 10
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were n.trees = 100, interaction.depth = 2, shrinkage = 0.1 and n.minobsinnode = 10.

$knn
k-Nearest Neighbors

356 samples
 13 predictor

Pre-processing: centered (13), scaled (13)
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 72, 70, 71, 72, 71
Resampling results across tuning parameters:

  k  RMSE      Rsquared   MAE
  5  6.091785  0.5728462  3.824682
  7  6.250168  0.5632543  3.991720
  9  6.365095  0.5651711  4.048343

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was k = 5.

attr(,"class")
[1] "caretList"
```

Ensemble Regression (R Code (caretEnsemble))

```
resamp <- resamples(model_list)
dotplot(resamp, metric = "RMSE")
modelCor(resamp)
                  lm  svmRadial          rf        gbm         knn
lm         1.00000000 -0.4306383 -0.04165515  0.5317569 -0.02440052
svmRadial -0.43063830  1.0000000  0.48667539  0.4688408 -0.63979779
rf        -0.04165515  0.4866754  1.00000000  0.6708609 -0.94916188
gbm        0.53175694  0.4688408  0.67086089  1.0000000 -0.80885772
knn       -0.02440052 -0.6397978 -0.94916188 -0.8088577  1.00000000
> ensemble_model <- caretEnsemble(model_list, metric = "RMSE" ,trControl = tc)
> summary(ensemble_model)
The following models were ensembled: lm, svmRadial, rf, gbm, knn
They were weighted:
-3.2969 0.2115 0.1444 0.9453 -0.1528 0.0214
The resulting RMSE is: 4.9036
The fit for each individual model on the RMSE is:
    method     RMSE    RMSESD
        lm 5.708080 0.4394401
 svmRadial 5.873460 0.4906549
        rf 4.611853 0.5572300
       gbm 5.684035 0.4209114
       knn 6.091785 0.2385785
> predict(ensemble_model , newdata = test_data)
```



Ensemble Regression (R Code (caretEnsemble))

```
> ensemble_model <- caretEnsemble(model_list, metric = "RMSE" ,trControl = tc )
> summary(ensemble_model)
The following models were ensembled: lm, svmRadial, rf, gbm, knn
They were weighted:
-3.2969 0.2115 0.1444 0.9453 -0.1528 0.0214
The resulting RMSE is: 4.9036
The fit for each individual model on the RMSE is:
    method     RMSE    RMSESD
        lm 5.708080 0.4394401
 svmRadial 5.873460 0.4906549
        rf 4.611853 0.5572300
       gbm 5.684035 0.4209114
       knn 6.091785 0.2385785
> prediction_data <-test_data
> prediction_data$pred <- predict(ensemble_model , newdata = test_data)
> RMSE(prediction_data$pred , prediction_data$medv)
[1] 3.599718
> prediction_data
       crim   zn indus chas    nox    rm   age    dis rad tax ptratio      b lstat medv     pred
1   0.00632 18.0  2.31    0 0.5380 6.575  65.2 4.0900   1 296    15.3 396.90  4.98 24.0 28.32520
2   0.02731  0.0  7.07    0 0.4690 6.421  78.9 4.9671   2 242    17.8 396.90  9.14 21.6 23.90114
5   0.06905  0.0  2.18    0 0.4580 7.147  54.2 6.0622   3 222    18.7 396.90  5.33 36.2 34.83408
10  0.17004 12.5  7.87    0 0.5240 6.004  85.9 6.5921   5 311    15.2 386.71 17.10 18.9 19.91365
18  0.78420  0.0  8.14    0 0.5380 5.990  81.7 4.2579   4 307    21.0 386.75 14.67 17.5 18.01929
19  0.80271  0.0  8.14    0 0.5380 5.456  36.6 3.7965   4 307    21.0 288.99 11.69 20.2 18.92013
24  0.98843  0.0  8.14    0 0.5380 5.813 100.0 4.0952   4 307    21.0 394.54 19.88 14.5 13.60853
29  0.77299  0.0  8.14    0 0.5380 6.495  94.4 4.4547   4 307    21.0 387.94 12.80 18.4 19.10051
30  1.00245  0.0  8.14    0 0.5380 6.674  87.3 4.2390   4 307    21.0 380.23 11.98 21.0 21.66768
35  1.61282  0.0  8.14    0 0.5380 6.096  96.9 3.7598   4 307    21.0 248.31 20.34 13.5 12.77569
```

Wrap up machine learning preprocess

Machine Learning Preprocess (1. Ideally Dataset)

ควรจะเป็น **1 row per 1 identity**

1 identity จะแตกต่างตาม industry เช่น 1 user id/row , 1 name/row , 1 phone/row , 1 bank account/1row

ตัวอย่าง ในที่นี้คือ 1 คน/ 1 row (หลักการ aggregate : นับวันที่ record ล่าสุด)

name	weight	height	gender	record_day
robert	70	175	M	20200201
robert	71	175	M	20200401
kate	58	165	F	20200101
kate	57	165	F	20200201
kate	59	165	F	20200401
Don	70	180	M	20200101
Don	69	180	M	20200401

name	weight	height	age	record_day
robert	71	175	M	20200401
kate	59	165	F	20200401
Don	69	180	M	20200401

Machine Learning Preprocess (1. Ideally Dataset)

สร้างตัวอย่าง Data Frame

```
> library(dplyr)
> name <- c('robert','robert','kate','kate','kate','Don','Don')
> weight <- c(70,71,58,57,59,70,69)
> height <- c(175,175,165,165,165,180,180)
> gender <- c('M','M','F','F','F','M','M')
> record_day <- c(20200201,20200401,20200101,20200201,20200401,20200101,20200401)
> example_data <- data.frame(name, weight, height,gender , record_day)
> example_data
    name weight height gender record_day
1 robert     70    175      M   20200201
2 robert     71    175      M   20200401
3   kate     58    165      F   20200101
4   kate     57    165      F   20200201
5   kate     59    165      F   20200401
6    Don     70    180      M   20200101
7    Don     69    180      M   20200401
```

Code ที่จะนำ record ล่าสุดมา record เดียว

```
> df_clean <- example_data %>%
+     group_by(name) %>%
+     arrange(desc(record_day)) %>%
+     mutate(rnk = row_number()) %>%
+     filter(rnk == 1)
> df_clean
# A tibble: 3 x 6
# Groups:   name [3]
  name   weight height gender record_day   rnk
  <fct>   <dbl>  <dbl> <fct>       <dbl> <int>
1 robert     71    175 M        20200401     1
2 kate       59    165 F        20200401     1
3 Don        69    180 M        20200401     1
```

Machine Learning Preprocess (2. Descriptive Statistic)

ก่อนที่จะนำไปสร้าง machine learning Model เราควรเข้าใจ data ของเรา เบื้องต้นก่อน

เราจะใช้ data mtcars
โดยมี column y (label) = mpg

```
> library(mlbench)
> data("mtcars")
> str(mtcars)
'data.frame':	32 obs. of  11 variables:
 $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
 $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
 $ disp: num  160 160 108 258 360 ...
 $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
 $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
 $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
 $ qsec: num  16.5 17 18.6 19.4 17 ...
 $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
 $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
 $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
 $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

Machine Learning Preprocess (2. Descriptive Statistic)

```
library(skimr)
skimmed_data <- skim(mtcars)
skimmed_data

-- Data Summary ------------------------
                           Values
Name                       mtcars
Number of rows             32
Number of columns          11
_______________________
Column type frequency:
  numeric                  11
_______________________
Group variables            None

-- Variable type: numeric ---------------------------------------------------------------------------------------------------
-------------------------------------------------
# A tibble: 11 x 11
   skim_variable n_missing complete_rate    mean      sd    p0    p25    p50    p75   p100 hist
 * <chr>             <int>         <dbl>   <dbl>   <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl> <chr>
 1 mpg                   0             1  20.1     6.03  10.4   15.4   19.2   22.8   33.9  ▃▅▇▃▂
 2 cyl                   0             1   6.19    1.79   4      4      6      8      8    ▆▁▃▁▇
 3 disp                  0             1 231.    124.    71.1  121.   196.   326    472    ▇▃▃▃▂
 4 hp                    0             1 147.     68.6   52     96.5  123    180    335    ▇▇▅▃▂
 5 drat                  0             1   3.60    0.535  2.76   3.08   3.70   3.92   4.93 ▇▃▅▃▂
 6 wt                    0             1   3.22    0.978  1.51   2.58   3.32   3.61   5.42 ▃▅▇▃▂
 7 qsec                  0             1  17.8     1.79  14.5   16.9   17.7   18.9   22.9  ▃▅▇▃▂
 8 vs                    0             1   0.438   0.504  0      0      0      1      1    ▇▁▁▁▇
 9 am                    0             1   0.406   0.499  0      0      0      1      1    ▇▁▁▁▇
10 gear                  0             1   3.69    0.738  3      3      4      4      5    ▇▁▆▁▂
11 carb                  0             1   2.81    1.62   1      2      2      4      8    ▇▆▂▁▁
```

สามารถใช้ library **skimr** เช็ค data เบื้องต้นได้

Machine Learning Preprocess (3. Missing Value)

```
library(skimr)
df <- mtcars
df[9:12,'hp'] <- NA
skim(df)
```

เราต้องจัดการ missing value ก่อน (ควรตัด observation ไป หรือว่า ควร impute)

ใน case นี้เราจงใจใส่ missing value เข้าไปเอง (row 9:12 , column hp)

```
-- Data Summary ------------------------
                           Values
Name                       df
Number of rows             32
Number of columns          11
_______________________
Column type frequency:
  numeric                  11
_______________________
Group variables            None

-- Variable type: numeric ---------------------------------------------------------------------------------------------------
--------------------------------------
# A tibble: 11 x 11
   skim_variable n_missing complete_rate    mean      sd    p0    p25    p50    p75   p100 hist
 * <chr>             <int>         <dbl>   <dbl>   <dbl> <dbl>  <dbl>  <dbl>  <dbl>  <dbl> <chr>
 1 mpg                   0         1      20.1     6.03  10.4   15.4   19.2   22.8   33.9  ▃▅▇▃▂
 2 cyl                   0         1       6.19    1.79   4      4      6      8      8    ▆▁▃▁▇
 3 disp                  0         1     231.    124.    71.1  121.   196.   326    472    ▇▃▃▃▂
 4 hp                    4         0.875 149.     72.2   52     96    132.   186.   335    ▇▇▆▃▂
 5 drat                  0         1       3.60    0.535  2.76   3.08   3.70   3.92   4.93 ▅▃▅▇▂
 6 wt                    0         1       3.22    0.978  1.51   2.58   3.32   3.61   5.42 ▂▃▇▅▂
 7 qsec                  0         1      17.8     1.79  14.5   16.9   17.7   18.9   22.9  ▂▅▇▅▂
 8 vs                    0         1       0.438   0.504  0      0      0      1      1    ▇▁▁▁▇
 9 am                    0         1       0.406   0.499  0      0      0      1      1    ▇▁▁▁▇
10 gear                  0         1       3.69    0.738  3      3      4      4      5    ▇▁▆▁▂
11 carb                  0         1       2.81    1.62   1      2      2      4      8    ▇▆▂▁▁
```

Machine Learning Preprocess (4. Missing Value imputation)

```
imputed_model <- preProcess(df[ , -which(names(df)=="mpg")] , method = "knnImpute")
df_imputed <- predict(imputed_model , newdata = df)
skim(df_imputed)
-- Data Summary ------------------------
                           Values
Name                       df_imputed
Number of rows             32
Number of columns          11
_______________________
Column type frequency:
  numeric                  11
_______________________
Group variables            None

-- Variable type: numeric ---------------------------------------------------------------------------------------
# A tibble: 11 x 11
   skim_variable n_missing complete_rate      mean    sd     p0    p25     p50    p75  p100 hist
 * <chr>             <int>         <dbl>     <dbl> <dbl>  <dbl>  <dbl>   <dbl>  <dbl> <dbl> <chr>
 1 mpg                   0             1  2.01e+ 1 6.03  10.4   15.4   19.2    22.8   33.9 ▅▇▅▂
 2 cyl                   0             1 -1.47e-17 1.    -1.22  -1.22  -0.105   1.01   1.01 ▇▂▂▇
 3 disp                  0             1 -9.08e-17 1.    -1.29  -0.887 -0.278   0.769  1.95 ▇▅▃▂
 4 hp                    0             1 -6.67e- 2 0.965 -1.34  -0.735 -0.520   0.429  2.58 ▇▅▃▂
 5 drat                  0             1 -2.92e-16 1     -1.56  -0.966  0.184   0.605  2.49 ▅▇▅▂
 6 wt                    0             1  4.68e-17 1     -1.74  -0.650  0.110   0.401  2.26 ▃▅▇▂
 7 qsec                  0             1  5.30e-16 1     -1.87  -0.535 -0.0776  0.588  2.83 ▃▅▇▂
 8 vs                    0             1  6.94e-18 1.00  -0.868 -0.868 -0.868   1.12   1.12 ▇▁▁▇
 9 am                    0             1  4.51e-17 1.00  -0.814 -0.814 -0.814   1.19   1.19 ▇▁▁▇
10 gear                  0             1 -3.47e-18 1.00  -0.932 -0.932  0.424   0.424  1.78 ▇▅▅▂
11 carb                  0             1  3.17e-17 1     -1.12  -0.503 -0.503   0.735  3.21 ▇▅▂▁

df_imputed[9:12,"hp"]
[1] -0.7597072 -0.7126028 -0.7126028  0.2488815
```

เรา impute โดยใช้ KNN impute
(Note ก่อนที่จะ impute , algorithm ทำการ normalized data อัตโนมัติ (center,scale))

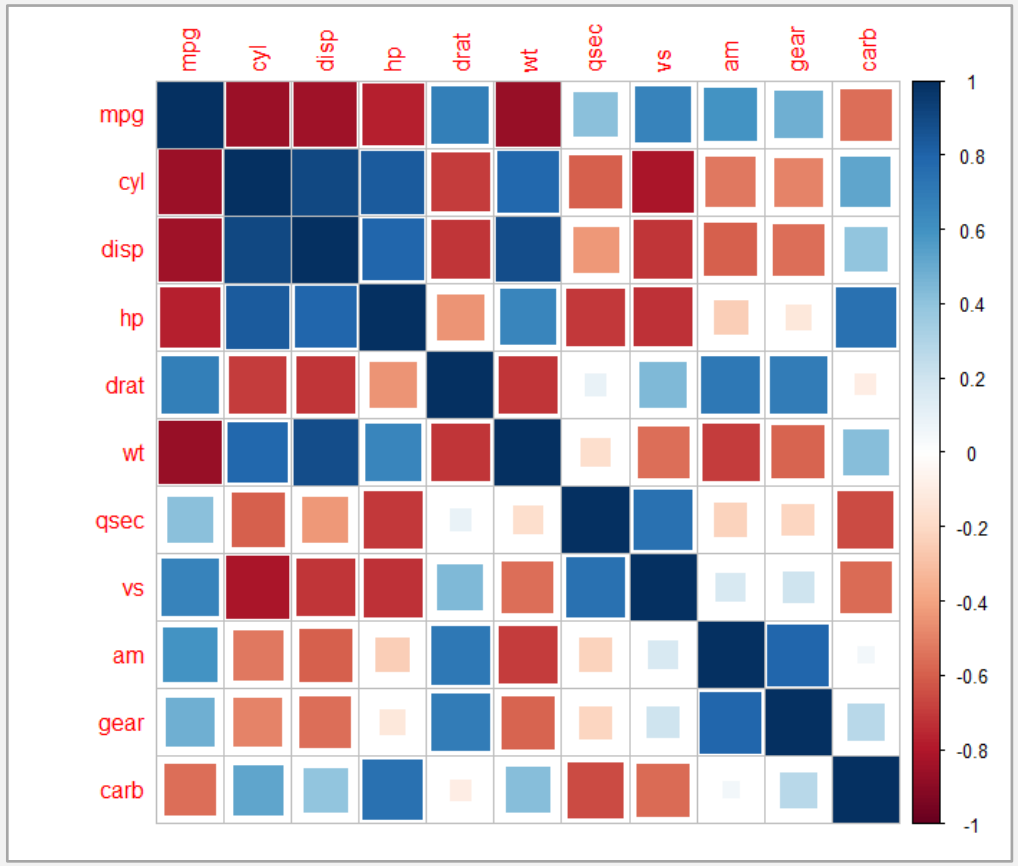
เราจะเห็นว่า data ที่ missing ถูกแทนค่า

Machine Learning Preprocess (5. correlation exploration)

```
library(corrplot)
df <- mtcars
corrplot(cor(df), method = "number")
corrplot(cor(df), method = "square")
```

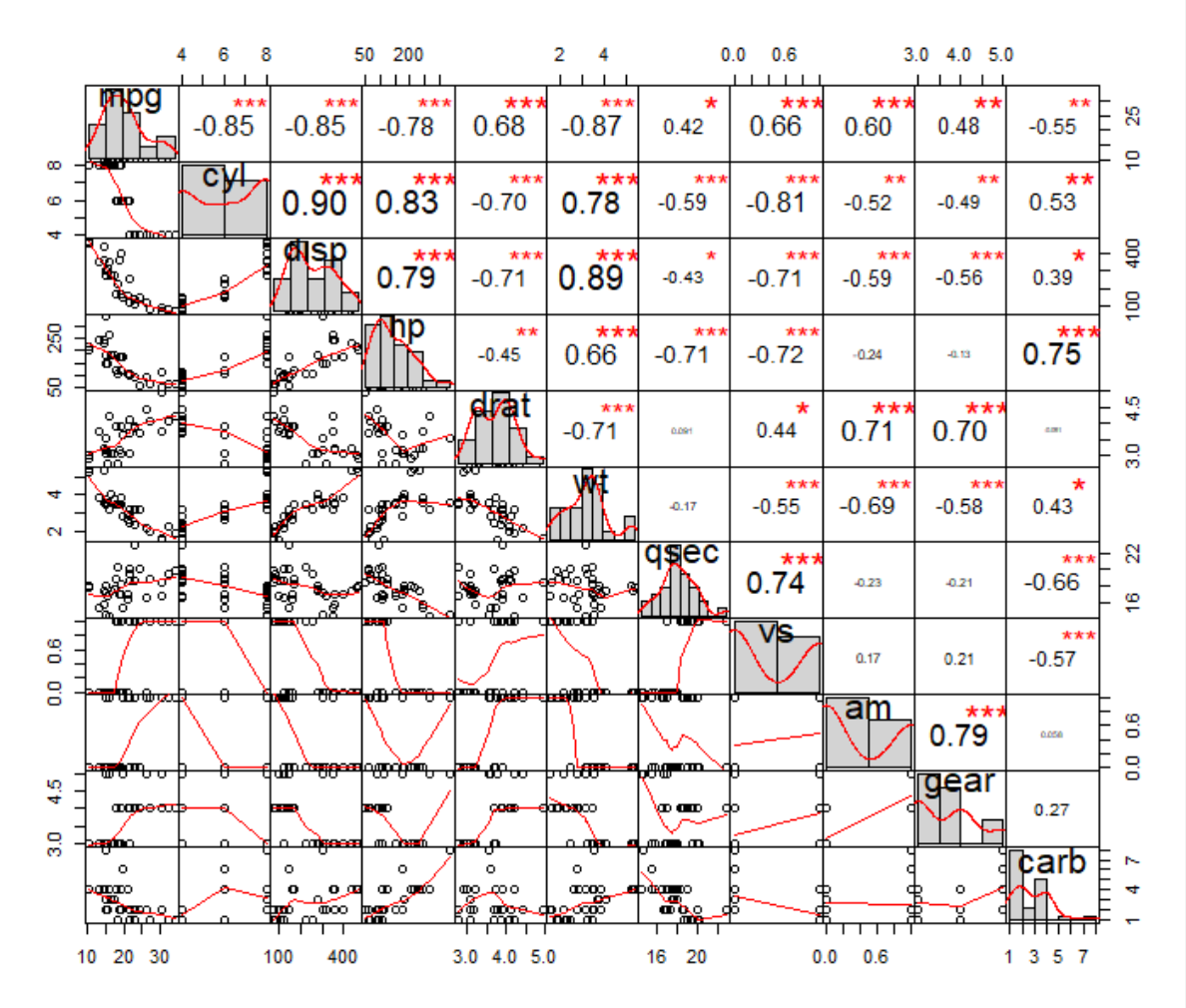
Explored correlation เพื่อ ดู feature คร่าวๆ ว่า

1. **Feature** correlated กับ **actual y** ไหม หรือ
2. **Feature** correlated กับ **feature** เองไหม



Machine Learning Preprocess (5. correlation exploration)

```
library(PerformanceAnalytics)

chart.Correlation(df)
```



Machine Learning Preprocess (6. split dataset)

```
set.seed(168)
df <- mtcars
num_train <- createDataPartition(y= df$mpg , p =0.7 , list = FALSE)
train_data <- df[num_train,]
test_data <- df[-num_train,]
str(train_data)
'data.frame':	24 obs. of  11 variables:
 $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
 $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
 $ disp: num  160 160 108 258 360 ...
 $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
 $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
 $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
 $ qsec: num  16.5 17 18.6 19.4 17 ...
 $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
 $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
 $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
 $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
str(test_data)
'data.frame':	8 obs. of  11 variables:
 $ mpg : num  16.4 32.4 30.4 21.5 15.2 19.2 15 21.4
 $ cyl : num  8 4 4 4 8 8 8 4
 $ disp: num  275.8 78.7 75.7 120.1 304 ...
 $ hp  : num  180 66 52 97 150 175 335 109
 $ drat: num  3.07 4.08 4.93 3.7 3.15 3.08 3.54 4.11
 $ wt  : num  4.07 2.2 1.61 2.46 3.44 ...
 $ qsec: num  17.4 19.5 18.5 20 17.3 ...
 $ vs  : num  0 1 1 1 0 0 0 1
 $ am  : num  0 1 1 0 0 0 1 1
 $ gear: num  3 4 4 3 3 3 5 4
 $ carb: num  3 1 2 1 2 2 8 2
```

แบ่ง Dataset เป็น 2 ก้อน train and test set

Proportion ในที่นี้คือ 70% training set , 30% testing set

Training set จะใช้ สร้าง model (และจะถูกแบ่งเป็น validation set)

Testing set จะใช้ evaluate model

Machine Learning Preprocess (7. scaling the dataset)

```
library(caret)
training_feature <- train_data[ , -which(names(train_data)=="mpg")]
preprocess_model <- preProcess(training_feature , method = c("center","scale"))
train_scaled <- predict(preprocess_model , newdata =train_data)
test_scaled <- predict(preprocess_model , newdata =test_data)
summary(train_scaled)
      mpg             cyl              disp               hp               drat               wt               qsec               vs               am
 Min.   :10.40   Min.   :-1.225   Min.   :-1.2879   Min.   :-1.3810   Min.   :-1.5646   Min.   :-1.7418   Min.   :-1.87401   Min.   :-0.868   Min.   :-0.8141
 1st Qu.:15.43   1st Qu.:-1.225   1st Qu.:-0.8867   1st Qu.:-0.7320   1st Qu.:-0.9661   1st Qu.:-0.6500   1st Qu.:-0.53513   1st Qu.:-0.868   1st Qu.:-0.8141
 Median :19.20   Median :-0.105   Median :-0.2777   Median :-0.3455   Median : 0.1841   Median : 0.1101   Median :-0.07765   Median :-0.868   Median :-0.8141
 Mean   :20.09   Mean   : 0.000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.000   Mean   : 0.0000
 3rd Qu.:22.80   3rd Qu.: 1.015   3rd Qu.: 0.7688   3rd Qu.: 0.4859   3rd Qu.: 0.6049   3rd Qu.: 0.4014   3rd Qu.: 0.58830   3rd Qu.: 1.116   3rd Qu.: 1.1899
 Max.   :33.90   Max.   : 1.015   Max.   : 1.9468   Max.   : 2.7466   Max.   : 2.4939   Max.   : 2.2553   Max.   : 2.82675   Max.   : 1.116   Max.   : 1.1899
      gear              carb
 Min.   :-0.9318   Min.   :-1.1222
 1st Qu.:-0.9318   1st Qu.:-0.5030
 Median : 0.4236   Median :-0.5030
 Mean   : 0.0000   Mean   : 0.0000
 3rd Qu.: 0.4236   3rd Qu.: 0.7352
 Max.   : 1.7789   Max.   : 3.2117
```

Scaling เพื่อให้ หน่วย unit ใกล้ๆกัน model จะได้ไม่ถูก influence โดย feature ใด feature นึง

Model ที่ใช้ scaling : Regularized regression , KNN regression , Support vector regression

Model ที่ไม่ใช้ scaling : Tree related regression , Linear regression

Machine Learning Preprocess (8. additional preprocessing)

```
library(caret)
training_feature <- train_data[ , -which(names(train_data)=="mpg")]
preprocess_model <- preProcess(training_feature , method = c("center","scale"))
train_scaled <- predict(preprocess_model , newdata =train_data)
test_scaled <- predict(preprocess_model , newdata =test_data)
summary(train_scaled)
```

1. "**range**" : ปรับ data ให้ unit อยู่ระหว่าง 0 -1
2. "**center**" : ปรับ data ให้ mean = 0
3. "**scale**" : นำ data มาหารด้วย standard deviation เพื่อ remove หน่วยของ data
4. "**Boxcox**" : ลด skewness เพื่อ ให้ data เป็น normality
5. "**Yeojohnson**" : เหมือน Boxcox แต่ ใช้กับ negative value ได้
6. "**expoTrans**" : Exponential Transformation ที่ใช้กับ negative value ได้
7. "**pca**" : replace ข้อมูล ด้วย ค่า principal component
8. "**Ica**" : replace ข้อมูลด้วยค่า independent component
9. "**spatialSign**" : ทำให้ Data เป็น unit circle

Machine Learning Preprocess (9. Training the model [randomforest])

```
library(caret)
library(ranger)

set.seed(168)
fitControl <- trainControl(
   method = "repeatedcv",
   number = 3,
   repeats = 3)
randomforest_model_caret <- train(mpg~.,
                                  data = train_scaled,
                                  method = "ranger",
                                  importance = "permutation",
                                  trControl = fitControl,
                                  tuneGrid = expand.grid(
                                    mtry = seq(2,7,1),
                                    splitrule = "variance",
                                    min.node.size = seq(2,14,2)
                                  ))
randomforest_model_caret
```

```
Random Forest

32 samples
10 predictors

No pre-processing
Resampling: Cross-Validated (3 fold, repeated 3 times)
Summary of sample sizes: 20, 22, 22, 21, 21, 22, ...
Resampling results across tuning parameters:
```

mtry	min.node.size	RMSE	Rsquared	MAE
2	2	2.268779	0.8879484	1.893751
2	4	2.293753	0.8876509	1.911574
2	6	2.404228	0.8753287	1.980986
2	8	2.560159	0.8654191	2.062738
2	10	2.764648	0.8455786	2.164259
2	12	2.923718	0.8318517	2.280690
2	14	3.095195	0.8188710	2.421840
3	2	2.206197	0.8943679	1.828921
3	4	2.235227	0.8908691	1.858327
3	6	2.322841	0.8840906	1.921041
3	8	2.463045	0.8694362	1.999175
3	10	2.649275	0.8558085	2.085473
3	12	2.784032	0.8455866	2.165907
3	14	2.923085	0.8283722	2.281347
4	2	2.172865	0.8975369	1.816382
4	4	2.246123	0.8900603	1.865021
4	6	2.301258	0.8819243	1.903626
4	8	2.415425	0.8740589	1.969687
4	10	2.529487	0.8634341	2.014946
4	12	2.707959	0.8478553	2.117907
4	14	2.831926	0.8387127	2.214307
5	2	2.198927	0.8943558	1.834170
5	4	2.226259	0.8903145	1.859275
5	6	2.257765	0.8869499	1.865311
5	8	2.368027	0.8773782	1.934942
5	10	2.524094	0.8608965	2.027606
5	12	2.675380	0.8491714	2.110744
5	14	2.806770	0.8385949	2.201302
6	2	2.189077	0.8963993	1.809768
6	4	2.224124	0.8921837	1.868404
6	6	2.283589	0.8851262	1.895103
6	8	2.383632	0.8752495	1.957968
6	10	2.513337	0.8654900	2.030956
6	12	2.648303	0.8517696	2.085374
6	14	2.765368	0.8437044	2.172404
7	2	2.206960	0.8920633	1.844229
7	4	2.216280	0.8918517	1.868573
7	6	2.297551	0.8851081	1.911586
7	8	2.394152	0.8744185	1.972932
7	10	2.550688	0.8598329	2.065862
7	12	2.658247	0.8481777	2.122442
7	14	2.786314	0.8410845	2.191762

```
Tuning parameter 'splitrule' was held constant at a value of variance
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were mtry = 4, splitrule = variance
and min.node.size = 2.
```

Machine Learning Preprocess (9. Training the model [Lasso regression])

```
library(caret)
library(glmnet)
set.seed(7891)
fitControl <- trainControl(
   method = "repeatedcv",
   number = 3,
   repeats = 3)
lasso_model_caret <-      train(mpg ~ .,
                                data = train_scaled,
                                method = "glmnet",
                                trControl = fitControl,
                                tuneGrid = expand.grid(alpha = 1,
                                                       lambda = seq(0.01 ,4 ,length=18)
                                ))
lasso_model_caret
```

```
glmnet

24 samples
10 predictors

No pre-processing
Resampling: Cross-Validated (3 fold, repeated 3 times)
Summary of sample sizes: 15, 16, 17, 16, 16, 16, ...
Resampling results across tuning parameters:

  lambda     RMSE      Rsquared   MAE
  0.0100000  5.228343  0.4958916  3.969866
  0.2447059  2.904251  0.8204618  2.356758
  0.4794118  2.735133  0.8510793  2.203706
  0.7141176  2.686237  0.8657962  2.164063
  0.9488235  2.687303  0.8765311  2.177006
  1.1835294  2.726971  0.8823281  2.222340
  1.4182353  2.806535  0.8855107  2.288373
  1.6529412  2.916995  0.8860715  2.367102
  1.8876471  3.049917  0.8853806  2.450209
  2.1223529  3.199324  0.8836877  2.544171
  2.3570588  3.363082  0.8814669  2.650598
  2.5917647  3.539770  0.8784317  2.769331
  2.8264706  3.728957  0.8739587  2.907999
  3.0611765  3.928791  0.8680514  3.071091
  3.2958824  4.133947  0.8618673  3.253259
  3.5305882  4.344125  0.8534293  3.437182
  3.7652941  4.552923  0.8455220  3.613560
  4.0000000  4.760929  0.8381789  3.783829

Tuning parameter 'alpha' was held constant at a value of 1
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were alpha = 1 and lambda = 0.7141176.
```

Machine Learning Preprocess (10. interpreting result [randomforest])

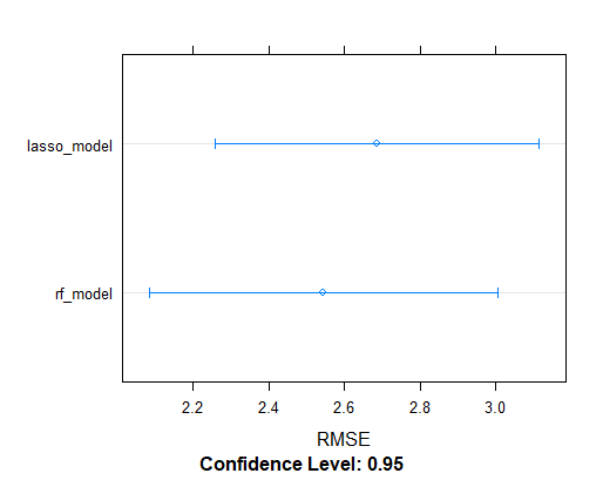
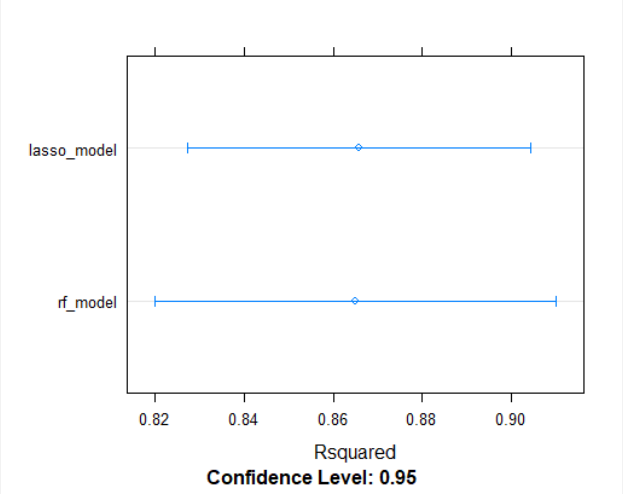
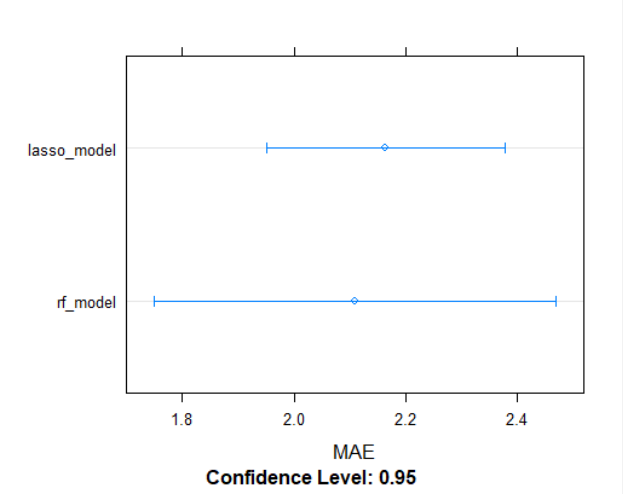
```
df_importance <- data.frame(randomforest_model_caret$finalModel$variable.importance)
names(df_importance) <- "importance_score"
df_importance[order(df_importance$importance_score , decreasing = TRUE),,drop = FALSE]
     importance_score
wt         9.29879049
cyl        8.39442108
disp       7.59862250
hp         6.04277745
carb       0.87046668
gear       0.52207673
am         0.26271518
drat       0.17592667
vs         0.16333344
qsec       0.04399769
```

Machine Learning Preprocess (10. interpreting result [Lasso regression])

```
coef(lasso_model_caret$finalModel, s = lasso_model_caret$bestTune$lambda)
11 x 1 sparse Matrix of class "dgCMatrix"
                     1
(Intercept) 19.64166667
cyl         -2.19417487
disp         .
hp          -0.21093910
drat         .
wt          -2.33029740
qsec         .
vs           .
am           .
gear         .
carb        -0.08948922
```

Machine Learning Preprocess (11. Prediction)

```
test_pred <- test_scaled
test_pred['pred_rf'] <- predict(randomforest_model_caret , newdata = test_scaled)
test_pred['pred_lasso'] <- predict(lasso_model_caret , newdata = test_scaled)
test_pred[c('mpg','pred_rf','pred_lasso')]
                  mpg  pred_rf pred_lasso
Merc 450SE       16.4 15.70441   15.48178
Fiat 128         32.4 27.65650   25.43352
Honda Civic      30.4 29.73210   26.75413
Toyota Corona    21.5 23.74380   24.72256
AMC Javelin      15.2 17.52635   17.09945
Pontiac Firebird 19.2 15.98486   16.07736
Maserati Bora    15.0 15.29911   15.77792
Volvo 142E       21.4 23.78630   23.89750
RMSE(test_pred$pred_rf , test_pred$mpg)
[1] 2.499957
RMSE(test_pred$pred_lasso , test_pred$mpg)
[1] 3.414194
```

Machine Learning Preprocess (12. Evaluation)

```
resamp <- resamples(list( rf_model = randomforest_model_caret ,
                          lasso_model =  lasso_model_caret))
dotplot(resamp , metric = "RMSE")
dotplot(resamp , metric = "Rsquared")
dotplot(resamp , metric = "MAE")
```