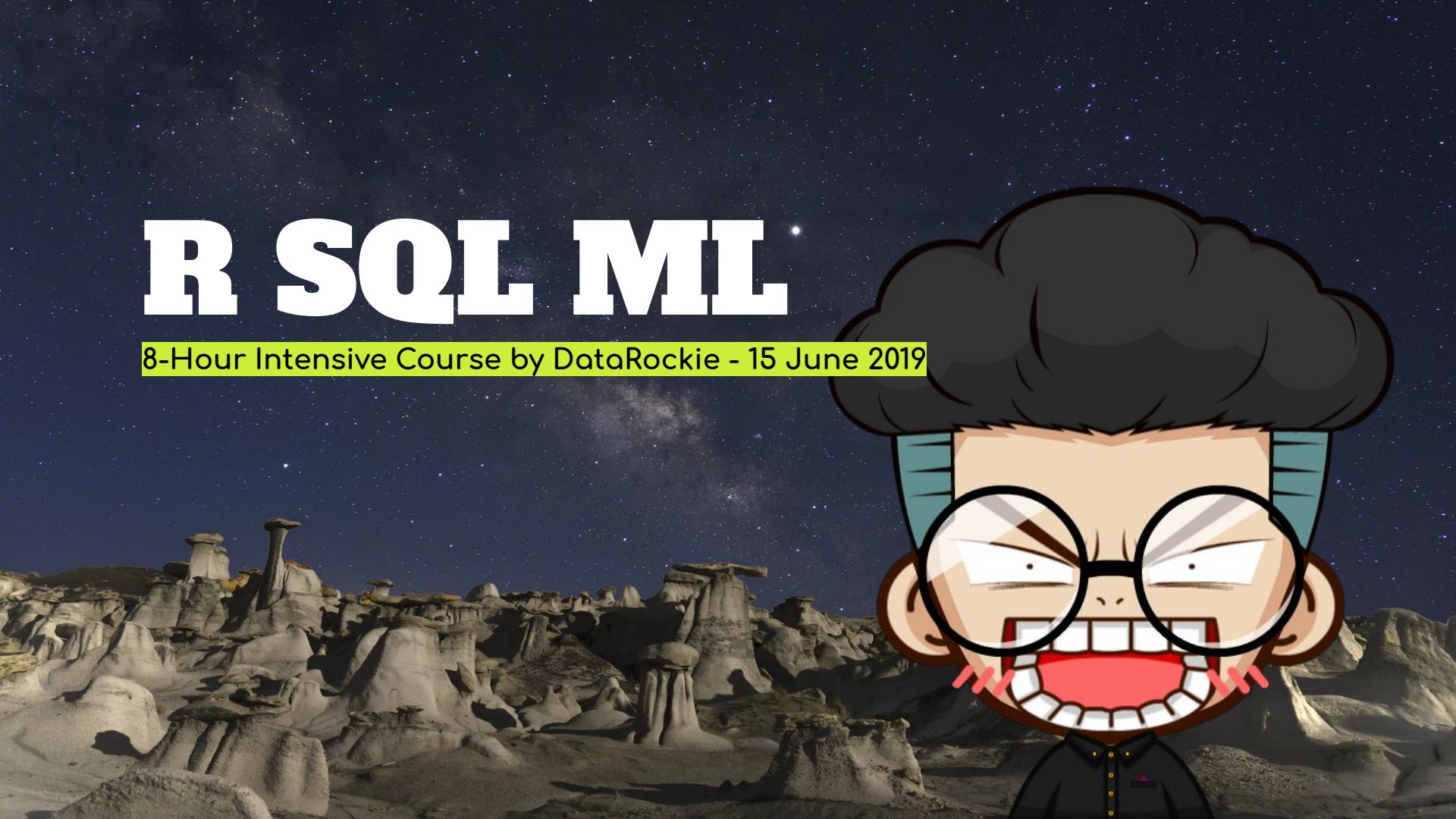
R SQL ML

8-Hour Intensive Course by DataRockie - 15 June 2019



Learning Resources

- Azure Notebook for This Course
  https://notebooks.azure.com/kasidistoy
- Google Drive for This Course
  http://bit.ly/RSQLMLdrive
- R Cheat Sheet by RStudio
  https://www.rstudio.com/resources/cheatsheets

นักเรียนต้องสมัครฟรี Azure Account ก่อนนะครับ :)

Why R?

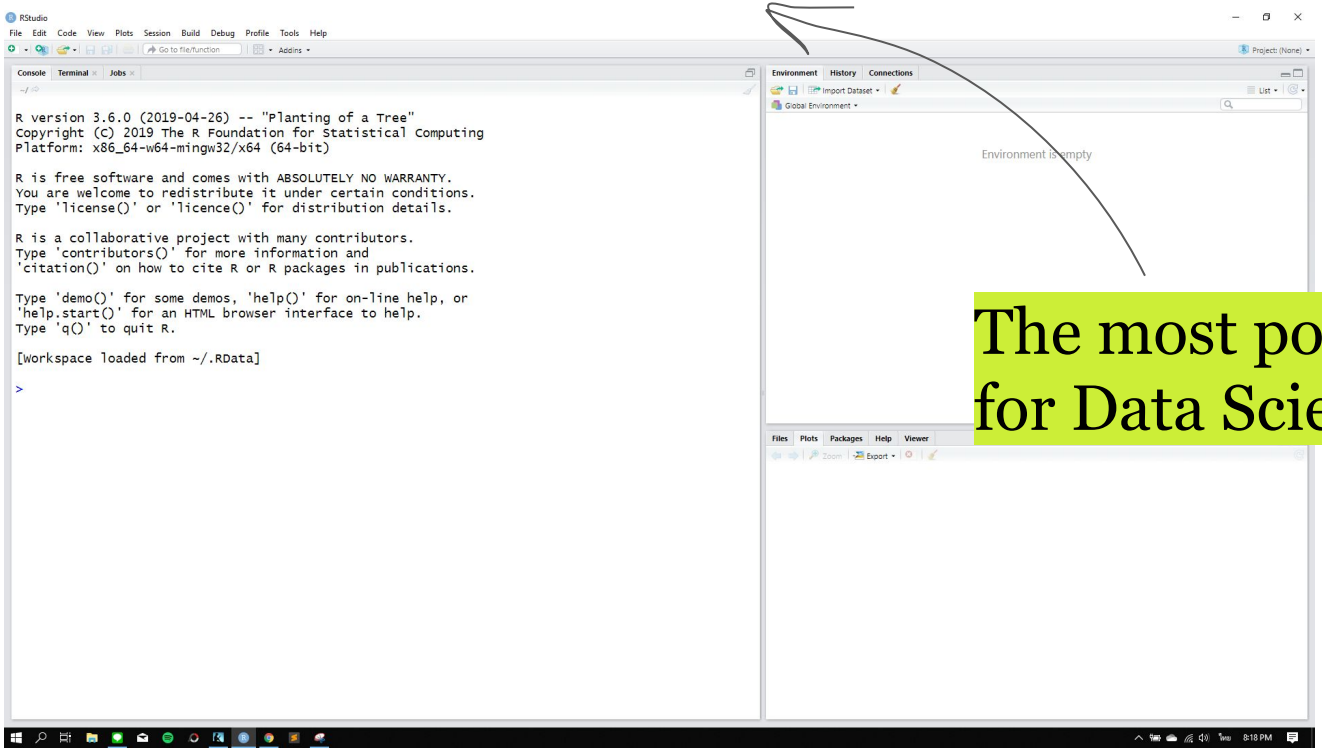
- Easy to Learn
- Powerful
- Top Language for Data Science

You can make a living

https://www.indeed.com/salaries/R-Programmer-Salaries

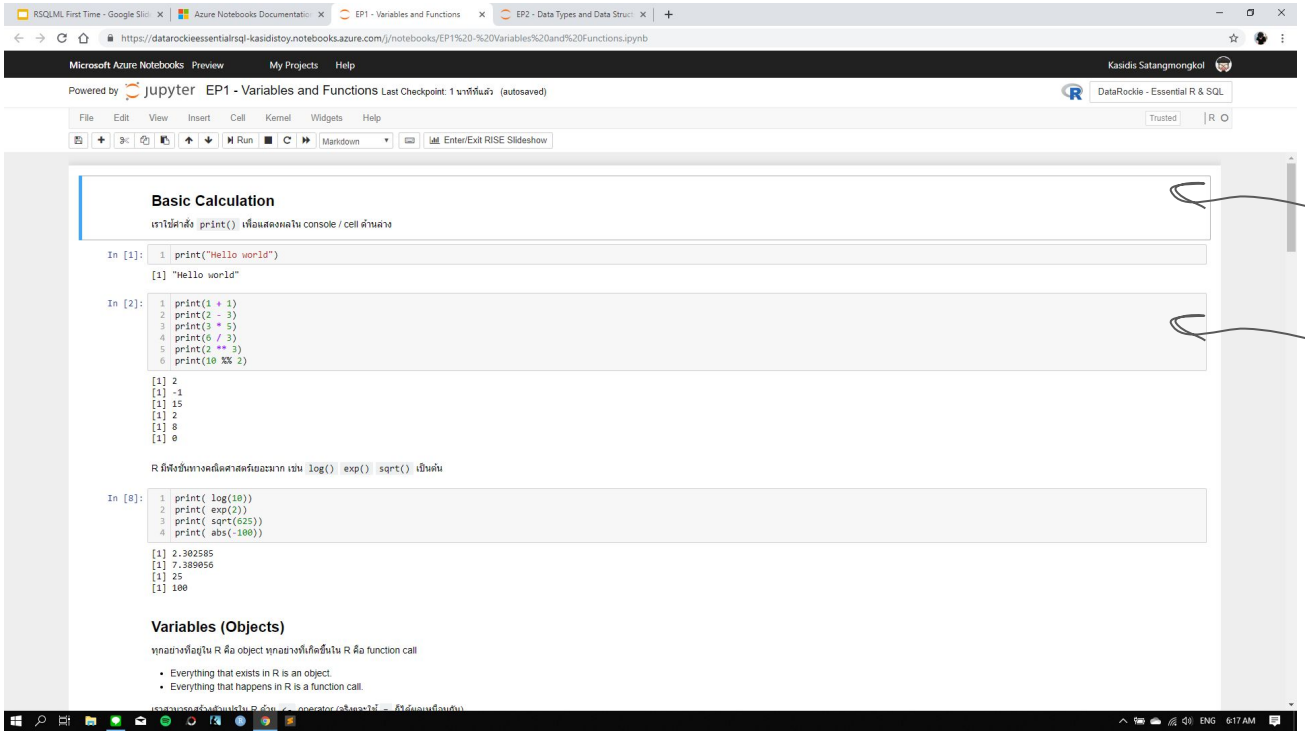
Programmer 2,270 salaries reported Programmer Jobs	**$73,990** per year	
Programmer Analyst 1,184 salaries reported Programmer Analyst Jobs	**$73,452** per year	
Data Scientist 3,152 salaries reported Data Scientist Jobs	**$120,301** per year	
Senior Programmer 370 salaries reported Senior Programmer Jobs	**$94,746** per year	
Data Analyst 3,840 salaries reported Data Analyst Jobs	**$65,502** per year	

RStudio 1.2



The most powerful IDE for Data Science

Azure (Jupyter) Notebooks



Text Cell

Code Cell

Bee

Top

Ink

Wan

Ing

Rut

Toy

MOTIVATION

ถ้าเราคิดว่า**ง่าย** มันก็**ง่าย**

End of today

- Be able to **read and write** R and SQL
- Be able to **train** ML models
- You can continue to learn by yourself


R
The most powerful + simple language for data analysis

Essential R

- variable
- data types
- data structures
- control flow
- function

Everything that exists in R is an **object**.
Everything that happens in R is a **function call**.

John Chambers

Founder of S and R core team

Calculation

```
1 + 1
2 * 2
3 / 5
10 - 20
5 %% 2

log(20)
exp(3)
sqrt(625)
abs(-100)
```

ตัวอย่างของฟังชั่นใน R

Variables

```
my_name <- “Toy”
my_age <- 31
wife <- FALSE

# This is a comment

# remove variables
rm(my_name)
rm(my_age)
```

Data Types

Type	Example
numeric	100, 200, 3.5, 5.9
character	“Hello”, “World”, “R”, “Python”
logical	TRUE, FALSE
factor	categorical data in statistics e.g. gender (M/F), animals (dog, cat)

Data Types

```
## c() is a function to create vector
friends <- c(“Top”, “Bee”, “Wan”, “Ink”, “Rut”)
ages <- c(26, 31, 25, 25, 25)
gender <- c(“Male”, “Male”, “Male”, “Male”, “Female”)
senior <- c(TRUE, TRUE, FALSE, FALSE, FALSE)

## check data types
class(friends)
class(ages)
class(gender)
class(senior)
```

Logical #1

```
## create variable x
x <- 100

## conditions
x == 100         ## TRUE
x != 100         ## FALSE
x > 20           ## TRUE
x < 300          ## TRUE
is.numeric(x)    ## TRUE
is.character(x) ## FALSE
```

Logical #2

```
x <- c(100, 200, 300, 400, 500)

x >= 300
## FALSE FALSE TRUE TRUE TRUE

sum(x >= 300)
## 3

mean(x >= 300)
## 0.6
```

TRUE มีค่าเท่ากับหนึ่ง
FALSE มีค่าเท่ากับศูนย์

Factor

```
## review factor
gender <- c(“Male”, “Male”, “Male”, “Male”, “Female”)
gender <- factor(gender)

## check data types
class(gender)
print(gender)
```

Data Structures

Type	Example
vector	c(100, 200, 300)
matrix	matrix(1:25, ncol = 5)
list	list(a = c(1,2,3), b = matrix(1:9))
dataframe	data.frame(col1, col2, col3)

[1] Vector

```
## there are so many ways to create vectors in R
x <- c(10, 20, 30, 99, 1000)

## use colon :
y <- 1:100

## use vector function such as seq(), rep()
z <- seq(1, 100, by = 0.5)
k <- rep(c(1,2,3), time = 10)
```

Vectorization

```
## this make R code runs very fast
x <- 1:10
x + 2
x * 2

y <- rep(2, time = 5)

## try this
x + y
```

Vectorization Explained

x	100	200	300	400	500
y	5	10	15	20	25

+

result	105	210	315	420	525

R works like Excel, But with (god) SPEED!

x	y	result	formula
100	5	105	=C3+D3
200	10	210	=C4+D4
300	15	315	=C5+D5
400	20	420	=C6+D6
500	25	525	=C7+D7

[2] Matrix

```
## matrix is very similar to vector
## but has 2 dimensions (row x column)
x <- 1:25
dim(x) <- c(5,5)
print(x)

## use matrix() function
x <- matrix(1:25, ncol = 5)
```

[3] Data Frame

```
## the most important data structure for data analysis
id <- 1:5
friends <- c(“Toy”, “Top”, “Bee”, “Ink”, “Wan”)
age <- c(31, 26, 31, 26, 25)
handsome <- c(F, T, T, T, T)

## create dataframe
df <- data.frame(id, friends, age, handsome)
```

[4] List

```
## think of list like your favourite playlist
rock <- c(“Incubus”, “Rolling Stones”)
pop <- c(“Ed Sheeran”, “Justin Bieber”)
hiphop <- c(“Jay Z”)
songs <- a_data_frame

## put them in a list
my_album <- list(rock, pop, hiphop, songs)
```

Quick Recap

Data Types
- numeric
- character
- logical
- factor

Data Structures
- vector
- matrix
- dataframe
- list

Index / Subset

```
## one dimension
x <- c(100, 200, 300, 99, 1000)
x[1]
x[1:3]
x[x > 200]

## two dimensions
mtcars[1,  ]
mtcars[ , 1]
mtcars[1:10,    ]
mtcars[    , 1:3]
mtcars[1:5 , 1:3]
mtcars[c(1,10,20), ]
mtcars[ , “mpg”]
```

What’s Function?

Input

Function

หน้าที่ของ function คือ map input เป็น output

Output

R Functions

function_name(arg1, arg2, ...)

argument คือ parameter ของฟังชั่นที่เราสามารถเปลี่ยนได้

```
## function sum
x <- 1:10
ans <- sum(x)
print(ans)
```

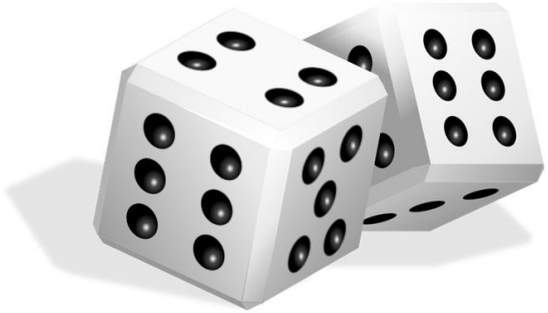
Function Template

```
function_name <- function(arguments) {
   … do something …
}
```

Your First Functions

```
## double function
double <- function(x) {
    x * 2
}

## triple function


## cube function
```

Casino Royale



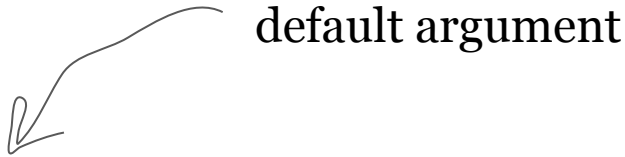
โจทย์: เขียนฟังชั่น roll_dices() หาผลรวมของการโยนลูกเต๋าสองลูก

```
## roll two dices and get total scores
## hint: use sample() function
sample(1:10, size = 1)

## function template
roll_dices <- function( ) {
    ... return total scores of two dices ...
}
```

Casino Royale (Solution)

```
roll_dices <- function( ) {
   die1 <- sample(1:6, 1)
   die2 <- sample(1:6, 1)
   die1 + die2
}
```

Casino Royale (Solution)




```
roll_dices <- function( n=2 ) {
   dies <- sample(1:6, size=n, replace=T)
   sum(dies)
}
```

ฟังชั่น roll_dices() ตอนนี้สามารถนับคะแนนลูกเต๋ากี่ลูกก็ได้

Statistical Functions

```
## working with dataframe
mtcars$mpg

## a lot of stats functions
mean( mtcars$mpg )
median( mtcars$mpg )
sd( mtcars$mpg )
var( mtcars$mpg )
min( mtcars$mpg )
max( mtcars$mpg )
cor( mtcars$mpg, mtcars$wt )
```

Analyzing DataFrame #1

```
## working with dataframe
head(mtcars)
tail(mtcars)
str(mtcars)
dim(mtcars)
summary(mtcars)
complete.cases(mtcars)

## its very easy to find column SUM and MEAN
colSums(mtcars)
colMeans(mtcars)
```

Analyzing DataFrame #2

```
## apply() function - one of the most useful functions in R
apply(mtcars, MARGIN = 1, mean)
apply(mtcars, MARGIN = 2, mean)
```

Function you want to apply

1 = Row-wise

2 = Column-wise

Analyzing DataFrame #3

```
## extract column from dataframe
mpg <- mtcars$mpg
summary(mpg)
```

```
summary(mpg)
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
10.40   15.43   19.20   20.09   22.80   33.90
```

Standardization (statistics)

$$\frac{X - \bar{X}}{SD}$$

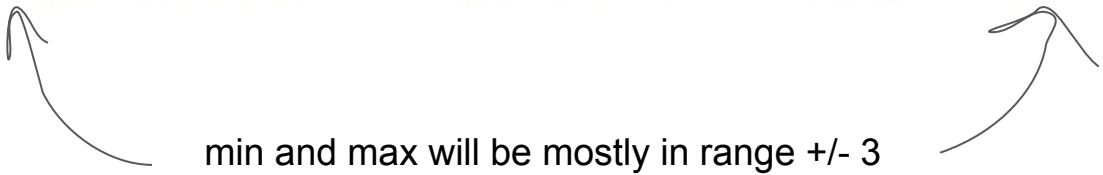
Mean

Standard Deviation

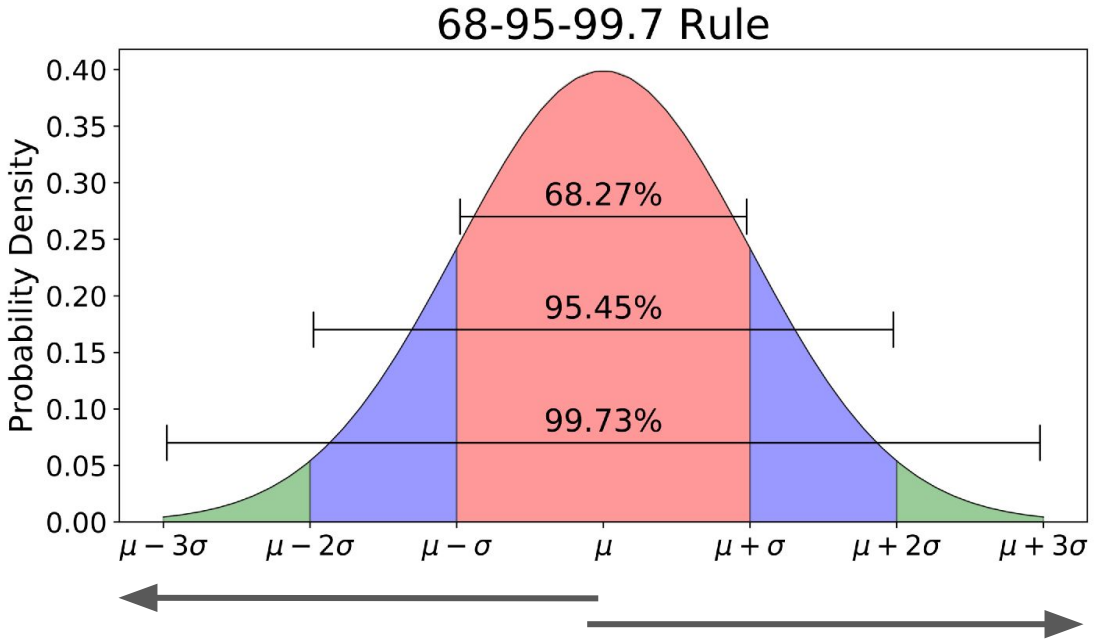
Analyzing DataFrame #3

```
## standardization (aka. center and scale)
mpg_norm <- (mpg - mean(mpg)) / sd(mpg)
summary(mpg_norm)
```

```
> summary(mpg_norm)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-1.6079 -0.7741 -0.1478  0.0000  0.4495  2.2913
```



Theory behind this function



+/- 3SD จากค่าเฉลี่ยตรงกลาง (Z score)

https://towardsdatascience.com/understanding-the-68-95-99-7-rule-for-a-normal-distribution-b7b7cbf760c2

Analyzing DataFrame #3

```
## we can create a function
standardize <- function(x) {
    (x - mean(x)) / sd(x)
}

## test function
mpg <- standardize(mtcars$mpg)
summary(mpg)

## apply to all columns
mtcars_scale <- apply(mtcars, 2, standardize)
```

Function เป็นโค้ดที่สามารถใช้ซ้ำได้
(Reusable piece of code)

Do it yourself

โจทย์: จงเขียนฟังชั่น feature_scaling() เพื่อปรับค่า X ให้มีค่าอยู่ระหว่าง [0, 1] ด้วยสูตรด้านล่าง

$$\frac{X - \min(X)}{\max(X) - \min(X)}$$

Let’s Recap

- function is very powerful in every programming language
- it’s particularly important in R
- we can write our own functions to analyze data e.g. standardize(), feature_scaling()

Before we move on

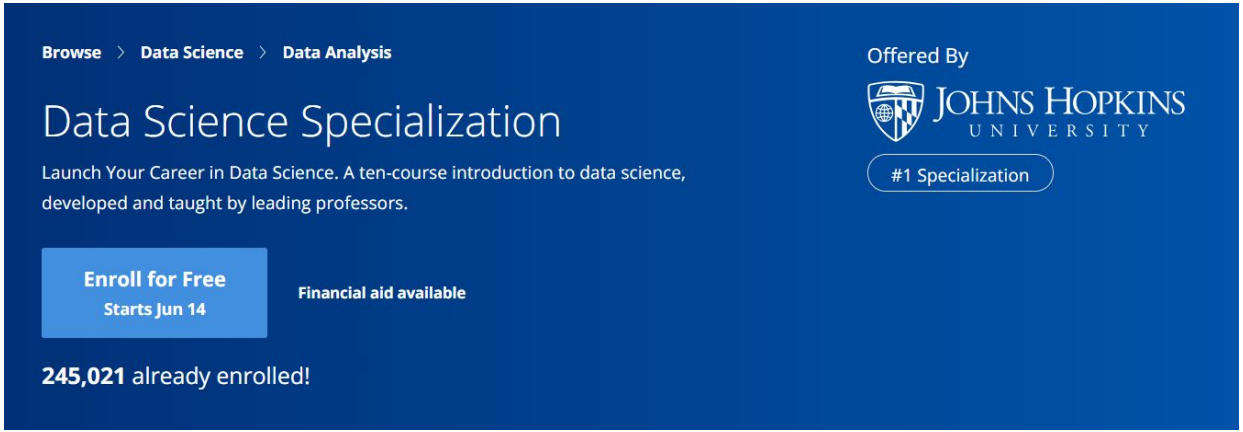
- make sure you have installed these two packages in **RStudio** (a lot of functions)

```
install.packages(“tidyverse”, dependencies=TRUE)
install.packages(“caret”, dependencies=TRUE)

library(tidyverse)
library(caret)
```

Practical Tips/ Recipe

```
## load library
library(tidyverse)

## read data into RStudio
df <- read.csv("datafile.csv")

## explore dataset
glimpse(df)
summary(df)
mean(complete.cases(df))

## rescale variables
feature_scaling <- function(x) (x-min(x))/ (max(x)-min(x))
df_scaled <- apply(df, 2, feature_scaling)
summary(df_scaled)
```

Learn More



https://www.coursera.org/specializations/jhu-data-science


TV
14


SQL
The standard language to talk to database

Essential SQL

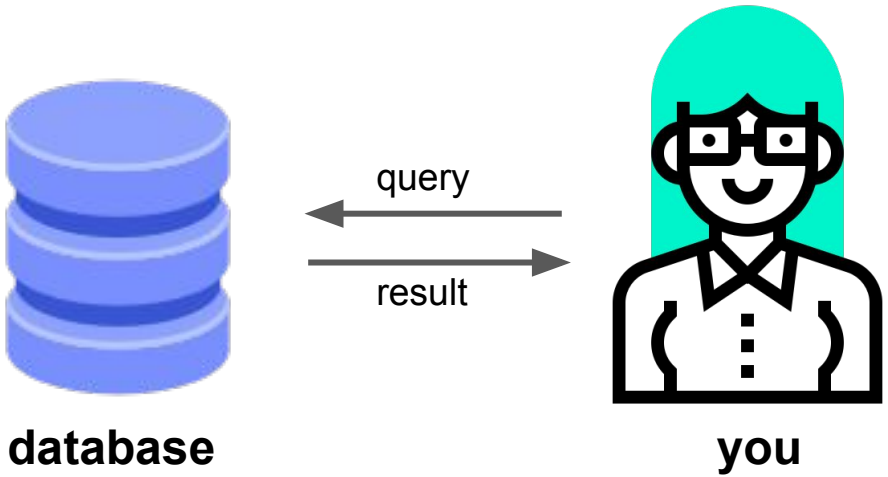
- select.. from..
- where
- having
- group by
- order by
- join

What’s SQL?

Structured (English) **Q**uery **L**anguage

ว. ถามคำถาม

How it works?



Relaxed! It’s plain English

```
SELECT * FROM mtcars;
```

เลือกทุกคอลั่มจาก table mtcars

Relaxed! It’s plain English

```
SELECT * FROM mtcars
WHERE hp < 100;
```

เลือกทุกคอลั่มจาก table mtcars ฟิลเตอร์เฉพาะที่ hp < 100

Relaxed! It’s plain English

```sql
SELECT mpg, wt, hp
FROM mtcars
WHERE hp < 100;
```

เลือกคอลั่ม mpg, hp, wt จาก table mtcars ฟิลเตอร์เฉพาะที่ hp < 100

Let’s write query in R

```
## get the package from R repository
install.packages(“sqldf”)
library(sqldf)

## write sql query in R
query <- “SELECT * FROM mtcars;”
result <- sqldf(query)
```

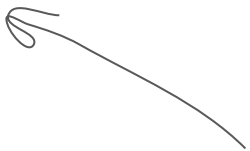
Extend your filters

```
SELECT mpg, wt, hp
FROM mtcars
WHERE hp < 100 AND am = 0;
```

```
## let's wrap this query with sqldf()
result <- sqldf("SELECT mpg, wt, hp FROM mtcars
                 WHERE hp < 100 AND am = 0;")
```

Aggregate Functions

```
SELECT
   COUNT(mpg),
   AVG(mpg),
   SUM(mpg),
   MIN(mpg),
   MAX(mpg)
FROM mtcars;
```



สรุปผลสถิติเบื้องต้น

Aggregate + Group By

```
SELECT
   AVG(mpg),
   SUM(mpg),
   am
FROM mtcars
GROUP BY am;
```

สรุปผลสถิติ avg, sum
แบ่งตามกลุ่ม am

```
> sqldf("SELECT
+       AVG(mpg),
+       SUM(mpg),
+       am
+       FROM mtcars
+       GROUP BY am;"
+ )
  AVG(mpg) SUM(mpg) am
1 17.14737     325.8  0
2 24.39231     317.1  1
```

Having to filter group

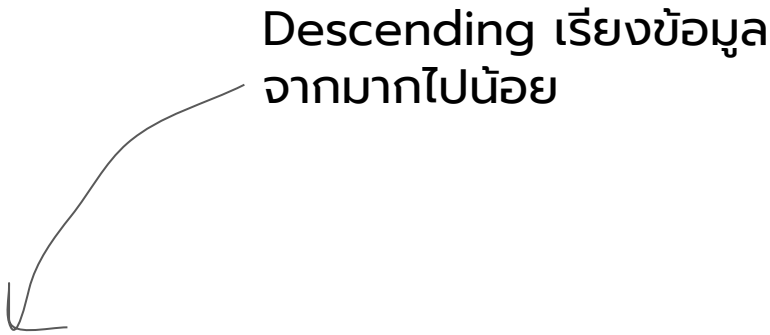
```sql
SELECT
   AVG(mpg),
   am
FROM mtcars
GROUP BY am
HAVING AVG(mpg) < 20;
```

Important: having ต้องมาหลัง group by เสมอ

Order By

```sql
SELECT mpg
FROM mtcars
ORDER BY mpg;
```

```sql
SELECT mpg
FROM mtcars
ORDER BY mpg DESC;
```



R Connect to SQLite

https://db.rstudio.com/databases/sqlite/

Function	What it does
`dbConnect()`	สร้าง connection ไปที่ database
`dbListTables()`	เรียกดูชื่อ tables ทั้งหมดใน database
`dbListFields()`	เรียกดูชื่อ columns ทั้งหมดใน table
`dbGetQuery()`	ดึงข้อมูลจาก database
`dbDisconnect()`	ปิด connection

R Connect to SQLite

https://db.rstudio.com/databases/sqlite/

```
## install packages
install.packages("RSQLite")
library(RSQLite)

## create connection and get data
con <- dbConnect(SQLite(), "chinook.db")
dbListTables(con)
dbListFields(con, "artists")
dbGetQuery(con, "SELECT * FROM customers LIMIT 10;")
dbDisconnect(con)
```

R Connect to MySQL

https://db.rstudio.com/databases/my-sql/

RMySQL.r*

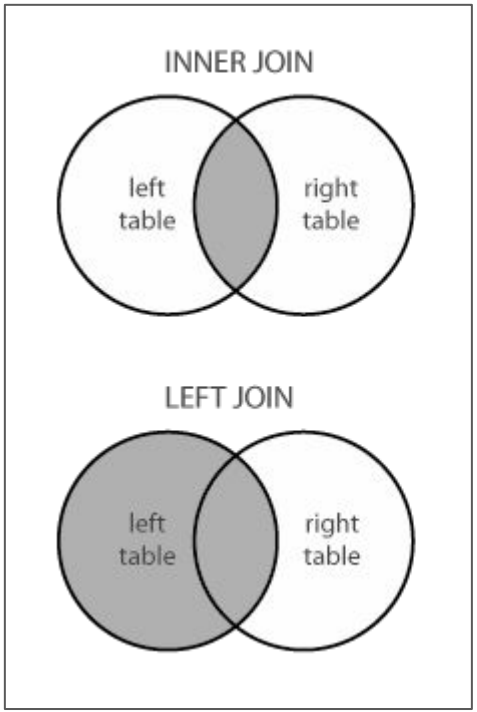
Source on Save

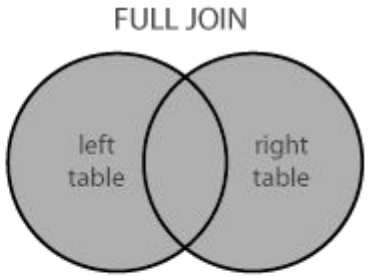
```r
library(RMySQL)
drv <- dbDriver("MySQL")
con <- dbConnect(drv,
                 username = "test",
                 password = "12345678",
                 port = 3306,
                 host = "localhost",
                 dbname = "sakila")

dbListTables(con)
df <- suppressWarnings(dbGetQuery(con, "SELECT * FROM actor LIMIT 20"))

dbDisconnect(con)
```

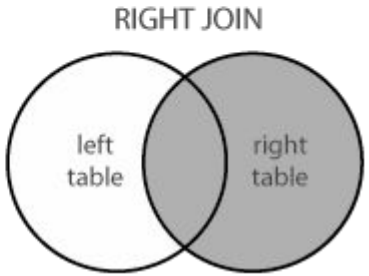
username/ password ต่อเข้า database ของเรา

Work with multiple tables

Most common joins







Inner Join

table1

id	name
1	Toy
2	Top
3	Bee
4	Ink
5	Wan

table2

id	department
1	Marketing
2	Data Science
5	Marketing
7	Finance
8	People Group

result

id	name	department
1	Toy	Marketing
2	Top	Data Science
5	Wan	Marketing

Left Join

table1

id	name
1	Toy
2	Top
3	Bee
4	Ink
5	Wan

table2

id	department
1	Marketing
2	Data Science
5	Marketing
7	Finance
8	People Group

result

id	name	department
1	Toy	Marketing
2	Top	Data Science
3	Bee	NULL
4	Ink	NULL
5	Wan	Marketing

Full Join

table1

id	name
1	Toy
2	Top
3	Bee
4	Ink
5	Wan

table2

id	department
1	Marketing
2	Data Science
5	Marketing
7	Finance
8	People Group

result

id	name	department
1	Toy	Marketing
2	Top	Data Science
3	Bee	NULL
4	Ink	NULL
5	Wan	Marketing
7	NULL	Finance
8	NULL	People Group

อันนี้นานๆใช้ที ไม่บ่อยเท่า inner/ left join

Example Code

inner join

```sql
SELECT
    table1.id,
    table1.name,
    table2.department
FROM table1
INNER JOIN table2
ON table1.id = table2.id;
```

left join

```sql
SELECT
    table1.id,
    table1.name,
    table2.department
FROM table1
LEFT JOIN table2
ON table1.id = table2.id;
```

Structured You Remember

```
SELECT ...
FROM ...
WHERE ...
GROUP BY ...
HAVING ...
ORDER BY ...
LIMIT ...
```

```
SELECT ...
FROM ...
INNER JOIN ...
ON ...
WHERE ...
GROUP BY ...
HAVING ...
ORDER BY ...
LIMIT ...
```

SQL Query ต้องเขียน
ตามลำดับนี้เลย

Learn More

Instructor

Sadie St. Lawrence

Data Scientist at VSP Global

Founder and CEO Women in Data (WID)

https://www.coursera.org/learn/sql-for-data-science

Data Wrangling

The most important skill of data analyst

The most useful package in R

dplyr package

- select()
- filter()
- arrange()
- mutate()
- summarise()
- group_by()

6 ฟังชั่นในตำนาน

Hadley Wickham

Summary (dplyr)

Function	What it does
select()	เลือกคอลั่มที่เราต้องการ
filter()	ฟิลเตอร์แถวที่เราต้องการ (ด้วยเงื่อนไข)
arrange()	เรียงข้อมูล
mutate()	สร้างคอลั่มใหม่
summarise()	สรุปผลสถิติ
group_by()	จับกลุ่มข้อมูล (นิยมใช้คู่กับ summarise())

Let’s practice %>%

```
## R way of analysis
head(mtcars)
mtcars %>% head()

## select columns
select(mtcars, hp, mpg, wt)

mtcars %>% select(hp, mpg, wt)
```

ได้ผลเหมือนกัน

select columns

```
mtcars %>%
    select(mpg, hp, wt, am)

mtcars %>%
    select(1:5)

mtcars %>%
    select(starts_with(“m”))
```

filter rows

```
mtcars %>%
    filter(hp < 100)

mtcars %>%
    filter(hp < 100 & am == 0)

mtcars %>%
    filter(hp < 100 | am == 0)

mtcars %>%
    filter(hp %in% 100:150)
```

arrange (sorting)

```
mtcars %>%
    arrange(hp)

mtcars %>%
    arrange(desc(hp))
```

mutate new columns

```
## dplyr (modern) way
mtcars %>%
    mutate(hp_double = hp * 2,
           hp_log = log(hp) )

## traditional way
mtcars$hp_double <- mtcars$hp * 2
mtcars$hp_double <- log(mtcars$hp)
```

summarise (statistics)

```
## dplyr (modern) way
mtcars %>%
    summarise(avg_hp = mean(hp),
              sd_hp = sd(hp),
              n = n())

## traditional way
mean(mtcars$hp)
sd(mtcars$hp)
length(mtcars$hp)
```

summarise + group by

```
mtcars %>%
    group_by(am) %>%
    summarise(avg_hp = mean(hp),
              sd_hp = sd(hp))

mtcars %>%
    group_by(am, cyl) %>%
    summarise(avg_hp = mean(hp),
              sd_hp = sd(hp))
```

group more than 2 variables

Output

```
> mtcars %>%
+     group_by(am, cyl) %>%
+     summarise(avg_hp = mean(hp),
+               sd_hp = sd(hp))
# A tibble: 6 x 4
# Groups:   am [2]
     am   cyl avg_hp sd_hp
  <dbl> <dbl>  <dbl> <dbl>
1     0     4   84.7 19.7
2     0     6  115.   9.18
3     0     8  194.  33.4
4     1     4   81.9 22.7
5     1     6  132.  37.5
6     1     8  300.  50.2
```

Think of %>% as data steps

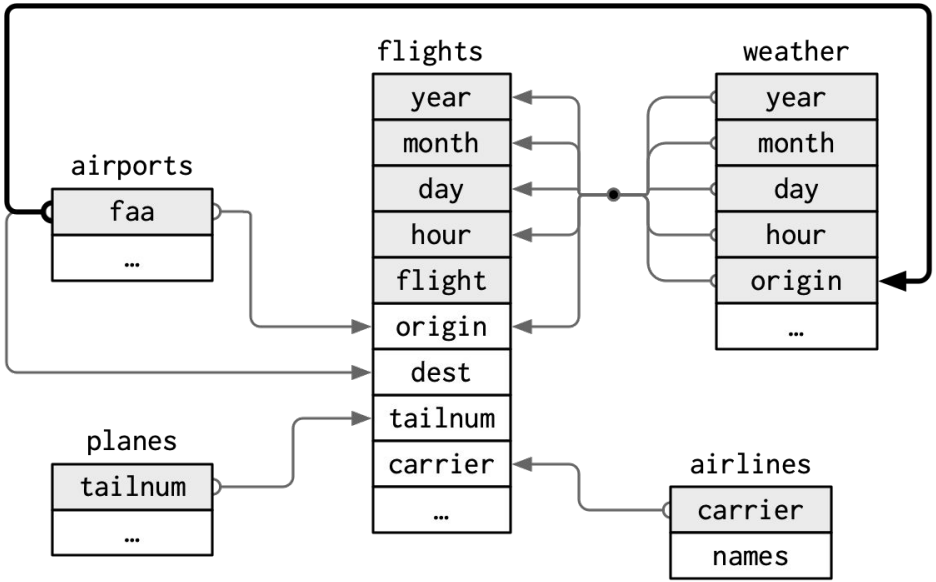
```
mtcars %>%
    select(mpg, hp, wt, am) %>%
    filter(am == 0) %>%
    mutate(hp_double = hp * 2) %>%
    summarise(avg_hp_double = mean(hp_double)) %>%
    arrange(desc(avg_hp_double))
```

step 1

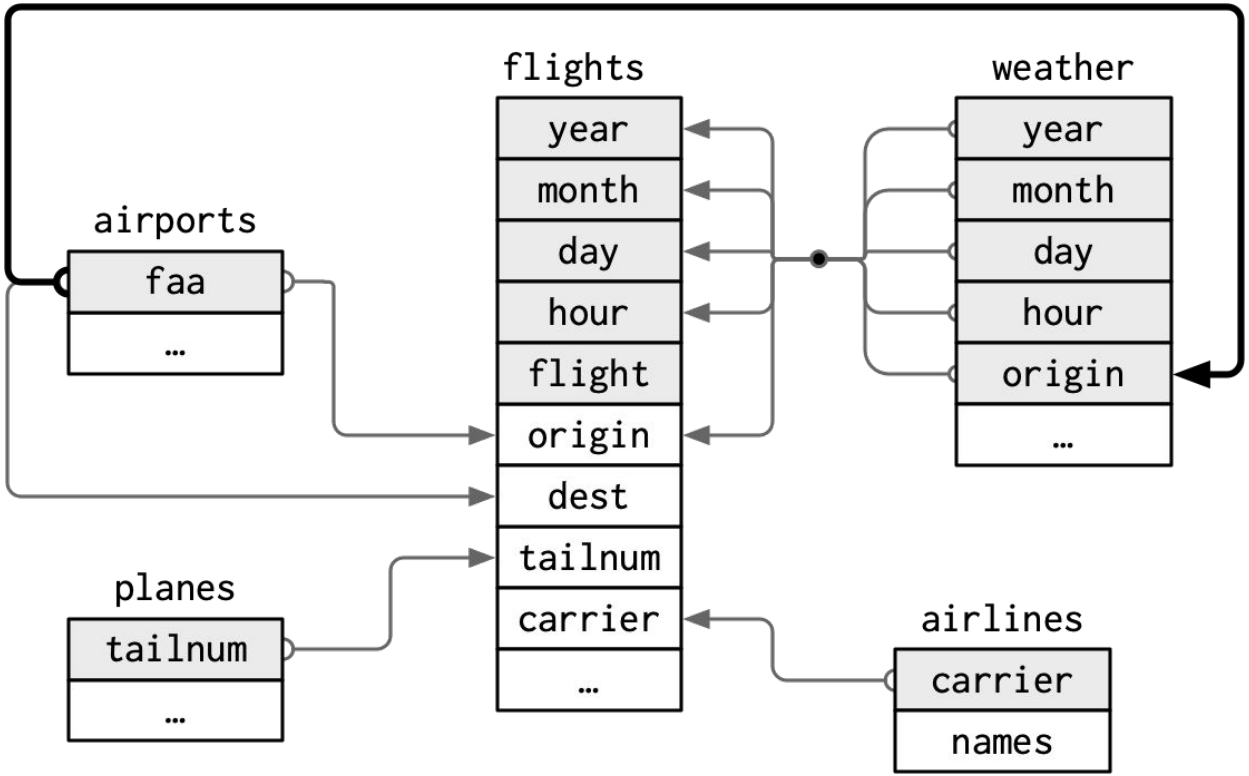
step 5

Work with multiple tables

```
install.packages("nycflights13")
library(nycflights13)
```



Relational Database



Let’s summarise data

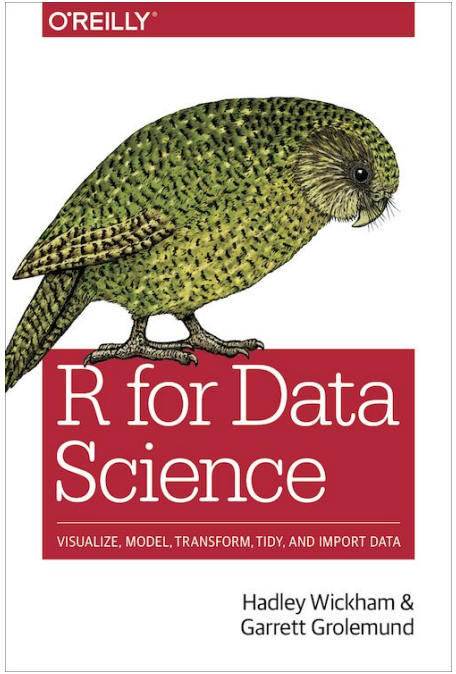
```
## explore dataset
library(tidyverse)
glimpse(flights)
glimpse(airlines)

## count frequency
flights %>% count(carrier)

top_ten <- flights %>%
    count(carrier, sort = TRUE) %>%
    head(10)
```

It’s super EASY in R

```
## let join the table with airlines
top_ten %>%
    left_join(airlines, by = “carrier”)
```

```
> top_10_carriers %>%
+     left_join(airlines, by = "carrier")
# A tibble: 10 x 3
   carrier     n name
   <chr>   <int> <chr>
 1 UA      58665 United Air Lines Inc.
 2 B6      54635 JetBlue Airways
 3 EV      54173 ExpressJet Airlines Inc.
 4 DL      48110 Delta Air Lines Inc.
 5 AA      32729 American Airlines Inc.
 6 MQ      26397 Envoy Air
 7 US      20536 US Airways Inc.
 8 9E      18460 Endeavor Air Inc.
 9 WN      12275 Southwest Airlines Co.
10 VX       5162 Virgin America
```

Learn More



Chapter 5 - data transformation

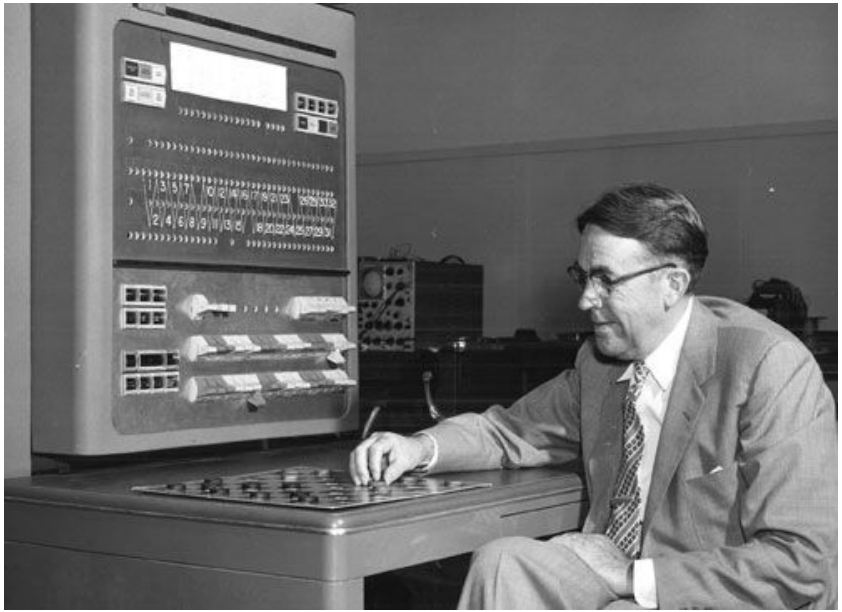
https://r4ds.had.co.nz/


Machine Learning
When a computer can learn to recognize pattern

Essential ML

- what exactly is machine learning
- supervised vs. unsupervised
- regression vs. classification
- train test split vs. cross validation
- model selection + hyperparameter
- model evaluation

Machine Learning



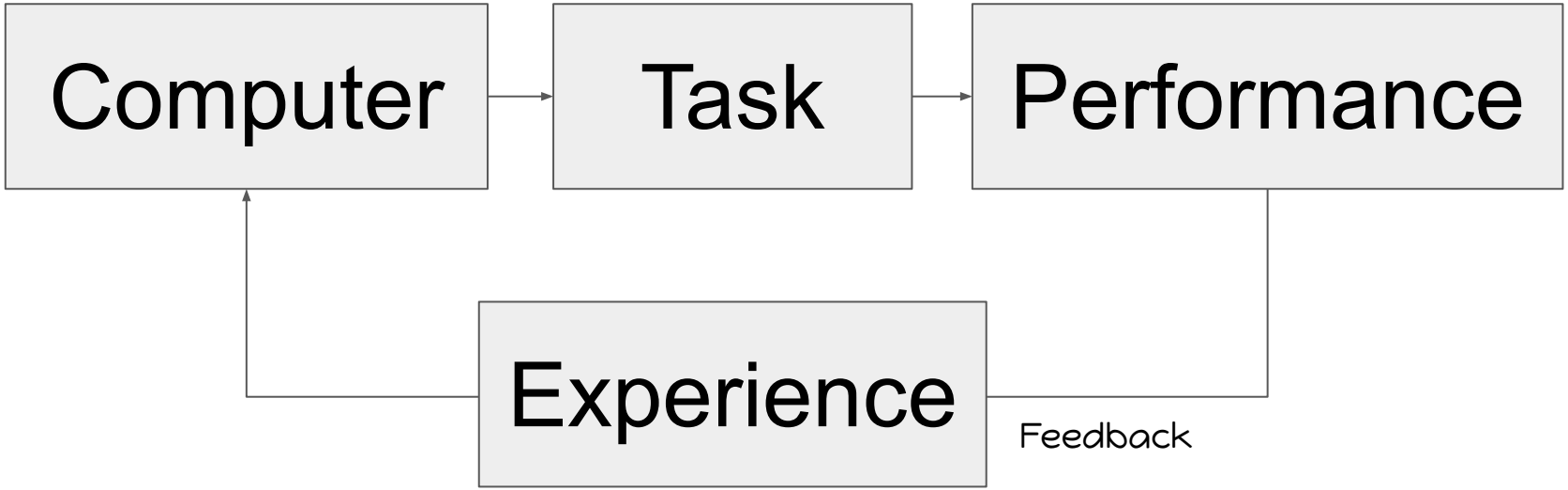
Arthur Samuel (1959)

Field of study that gives computers **the ability to learn without being explicitly programmed**.

How do we learn?

Human learn from experience.
Computer learn from data.

The Idea is Simple



ML glossary #1

- dataset
- data points
- features
- label or target

Data Point

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b	lstat	medv
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
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
3	0.02729	0.0	7.07	0	0.4690	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
4	0.03237	0.0	2.18	0	0.4580	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
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
6	0.02985	0.0	2.18	0	0.4580	6.430	58.7	6.0622	3	222	18.7	394.12	5.21	28.7
7	0.08829	12.5	7.87	0	0.5240	6.012	66.6	5.5605	5	311	15.2	395.60	12.43	22.9
8	0.14455	12.5	7.87	0	0.5240	6.172	96.1	5.9505	5	311	15.2	396.90	19.15	27.1
9	0.21124	12.5	7.87	0	0.5240	5.631	100.0	6.0821	5	311	15.2	386.63	29.93	16.5
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
11	0.22489	12.5	7.87	0	0.5240	6.377	94.3	6.3467	5	311	15.2	392.52	20.45	15.0
12	0.11747	12.5	7.87	0	0.5240	6.009	82.9	6.2267	5	311	15.2	396.90	13.27	18.9
13	0.09378	12.5	7.87	0	0.5240	5.889	39.0	5.4509	5	311	15.2	390.50	15.71	21.7
14	0.62976	0.0	8.14	0	0.5380	5.949	61.8	4.7075	4	307	21.0	396.90	8.26	20.4
15	0.63796	0.0	8.14	0	0.5380	6.096	84.5	4.4619	4	307	21.0	380.02	10.26	18.2

Dataset: BostonHousing

Features (X)

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b	lstat	medv
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
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
3	0.02729	0.0	7.07	0	0.4690	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
4	0.03237	0.0	2.18	0	0.4580	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
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
6	0.02985	0.0	2.18	0	0.4580	6.430	58.7	6.0622	3	222	18.7	394.12	5.21	28.7
7	0.08829	12.5	7.87	0	0.5240	6.012	66.6	5.5605	5	311	15.2	395.60	12.43	22.9
8	0.14455	12.5	7.87	0	0.5240	6.172	96.1	5.9505	5	311	15.2	396.90	19.15	27.1
9	0.21124	12.5	7.87	0	0.5240	5.631	100.0	6.0821	5	311	15.2	386.63	29.93	16.5
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
11	0.22489	12.5	7.87	0	0.5240	6.377	94.3	6.3467	5	311	15.2	392.52	20.45	15.0
12	0.11747	12.5	7.87	0	0.5240	6.009	82.9	6.2267	5	311	15.2	396.90	13.27	18.9
13	0.09378	12.5	7.87	0	0.5240	5.889	39.0	5.4509	5	311	15.2	390.50	15.71	21.7
14	0.62976	0.0	8.14	0	0.5380	5.949	61.8	4.7075	4	307	21.0	396.90	8.26	20.4
15	0.63796	0.0	8.14	0	0.5380	6.096	84.5	4.4619	4	307	21.0	380.02	10.26	18.2

Dataset: BostonHousing

Label / Target (Y)

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b	lstat	medv
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
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
3	0.02729	0.0	7.07	0	0.4690	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
4	0.03237	0.0	2.18	0	0.4580	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
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
6	0.02985	0.0	2.18	0	0.4580	6.430	58.7	6.0622	3	222	18.7	394.12	5.21	28.7
7	0.08829	12.5	7.87	0	0.5240	6.012	66.6	5.5605	5	311	15.2	395.60	12.43	22.9
8	0.14455	12.5	7.87	0	0.5240	6.172	96.1	5.9505	5	311	15.2	396.90	19.15	27.1
9	0.21124	12.5	7.87	0	0.5240	5.631	100.0	6.0821	5	311	15.2	386.63	29.93	16.5
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
11	0.22489	12.5	7.87	0	0.5240	6.377	94.3	6.3467	5	311	15.2	392.52	20.45	15.0
12	0.11747	12.5	7.87	0	0.5240	6.009	82.9	6.2267	5	311	15.2	396.90	13.27	18.9
13	0.09378	12.5	7.87	0	0.5240	5.889	39.0	5.4509	5	311	15.2	390.50	15.71	21.7
14	0.62976	0.0	8.14	0	0.5380	5.949	61.8	4.7075	4	307	21.0	396.90	8.26	20.4
15	0.63796	0.0	8.14	0	0.5380	6.096	84.5	4.4619	4	307	21.0	380.02	10.26	18.2

Dataset: BostonHousing

Features (X)

Label / Target (Y)

Supervised Learning

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b	lstat	medv
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
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
3	0.02729	0.0	7.07	0	0.4690	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
4	0.03237	0.0	2.18	0	0.4580	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
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
6	0.02985												5.21	28.7
7	0.08829												12.43	22.9
8	0.14455												19.15	27.1
9	0.21124	12.5	7.87	0	0.5240	5.631	100.0	6.0821	5	311	15.2	386.63	29.93	16.5
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
11	0.22489	12.5	7.87	0	0.5240	6.377	94.3	6.3467	5	311	15.2	392.52	20.45	15.0
12	0.11747	12.5	7.87	0	0.5240	6.009	82.9	6.2267	5	311	15.2	396.90	13.27	18.9
13	0.09378	12.5	7.87	0	0.5240	5.889	39.0	5.4509	5	311	15.2	390.50	15.71	21.7
14	0.62976	0.0	8.14	0	0.5380	5.949	61.8	4.7075	4	307	21.0	396.90	8.26	20.4
15	0.63796	0.0	8.14	0	0.5380	6.096	84.5	4.4619	4	307	21.0	380.02	10.26	18.2

Dataset: BostonHousing

Mapping

Features (X)

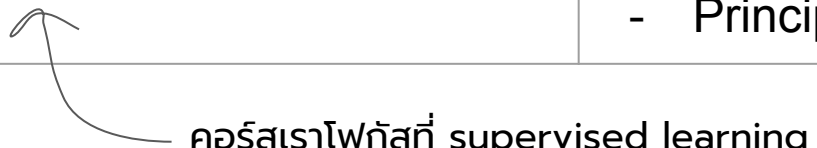
	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b	lstat
1	0.00632	18.0	2.31	0	0.5380	6.575	65.2	4.0900	1	296	15.3	396.90	4.98
2	0.02731	0.0	7.07	0	0.4690	6.421	78.9	4.9671	2	242	17.8	396.90	9.14
3	0.02729	0.0	7.07	0	0.4690	7.185	61.1	4.9671	2	242	17.8	392.83	4.03
4	0.03237	0.0	2.18	0	0.4580	6.998	45.8	6.0622	3	222	18.7	394.63	2.94
5	0.06905	0.0	2.18	0	0.4580	7.147	54.2	6.0622	3	222	18.7	396.90	5.33
6													
7													
8													
9	0.21124	12.5	7.87	0	0.5240	5.631	100.0	6.0821	5	311	15.2	386.63	29.93
10	0.17004	12.5	7.87	0	0.5240	6.004	85.9	6.5921	5	311	15.2	386.71	17.10
11	0.22489	12.5	7.87	0	0.5240	6.377	94.3	6.3467	5	311	15.2	392.52	20.45
12	0.11747	12.5	7.87	0	0.5240	6.009	82.9	6.2267	5	311	15.2	396.90	13.27
13	0.09378	12.5	7.87	0	0.5240	5.889	39.0	5.4509	5	311	15.2	390.50	15.71
14	0.62976	0.0	8.14	0	0.5380	5.949	61.8	4.7075	4	307	21.0	396.90	8.26
15	0.63796	0.0	8.14	0	0.5380	6.096	84.5	4.4619	4	307	21.0	380.02	10.26

Unsupervised Learning

Dataset: BostonHousing

Quick Summary

Supervised Learning	Unsupervised Learning
Has features (x) and labels (y)	Has features (x) without labels (y)
The goal is **PREDICT**	The goal is to **SUMMARISE**
Example algorithms - Regression - Classification	Example algorithms - Clustering - Association Rules - Principal Component Analysis



What problem is this?

AIS อยากจะทำ market survey กับลูกค้า (ทุกค่าย) ทั้งหมด 3000 คน เพื่อจะดูว่าตลาดคนไทยมีลูกค้าอยู่กี่ประเภท?
i.e. customer segmentation

What problem is this?

Gmail มีตัวกรอง email ว่าอันไหนคือ spam อันไหนคือ ham (อีเมล์ดี)

What problem is this?

อ้ิงเขียนโค้ดทำ web scraping จากเว็บไซต์ขายรถยนต์มือสอง เพื่อจะดูว่ารถยนต์ Toyota รุ่น 2015 เครื่อง 1.5 ลิตร ขับมาแล้ว 20000 โล ควรจะซื้อราคาเท่าไรดี?

Types of Supervised L.

1. Regression	2. Classification
Predict **numeric** labels	Predict **categorical** labels
Examples - house price - customer satisfaction - personal income - how much a customer will spend	Examples - yes/ no question - churn prediction - conversion - weather forecast - default prediction
100, 200, 250, 190, 300, 500, etc.	0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, etc.

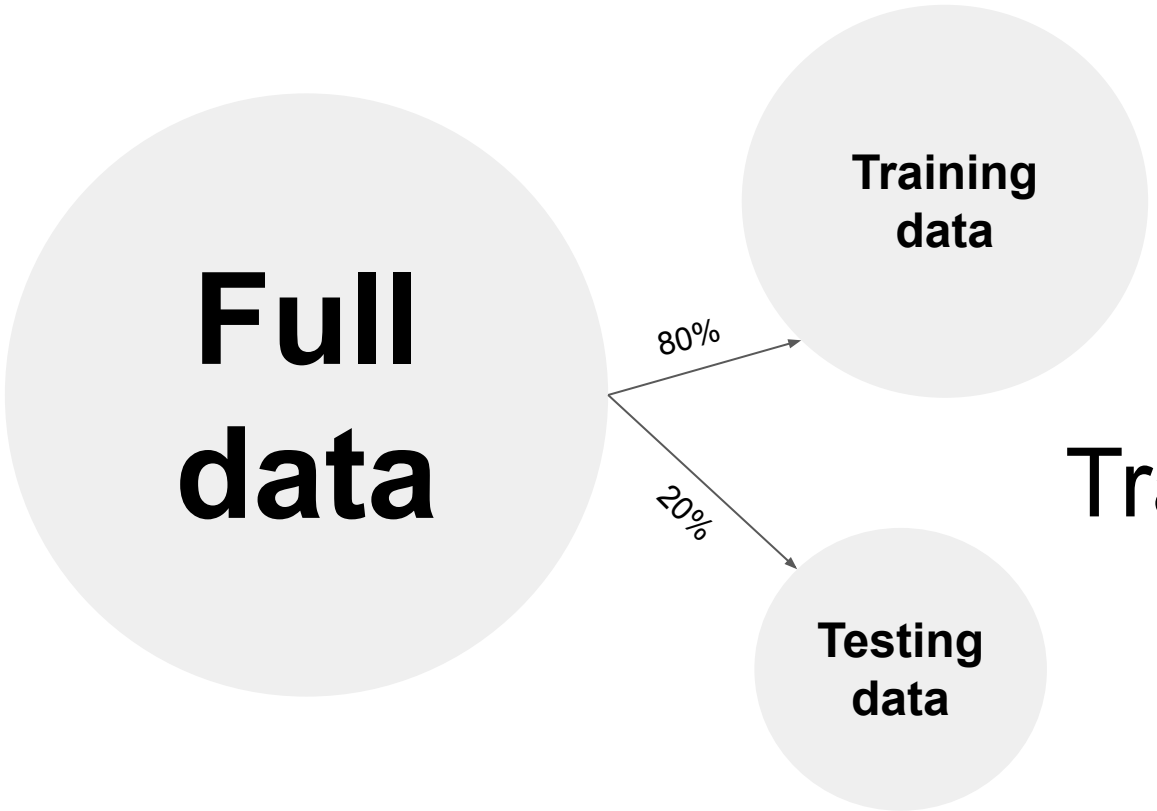
Now let's get into the details :)

3 Steps to Build ML

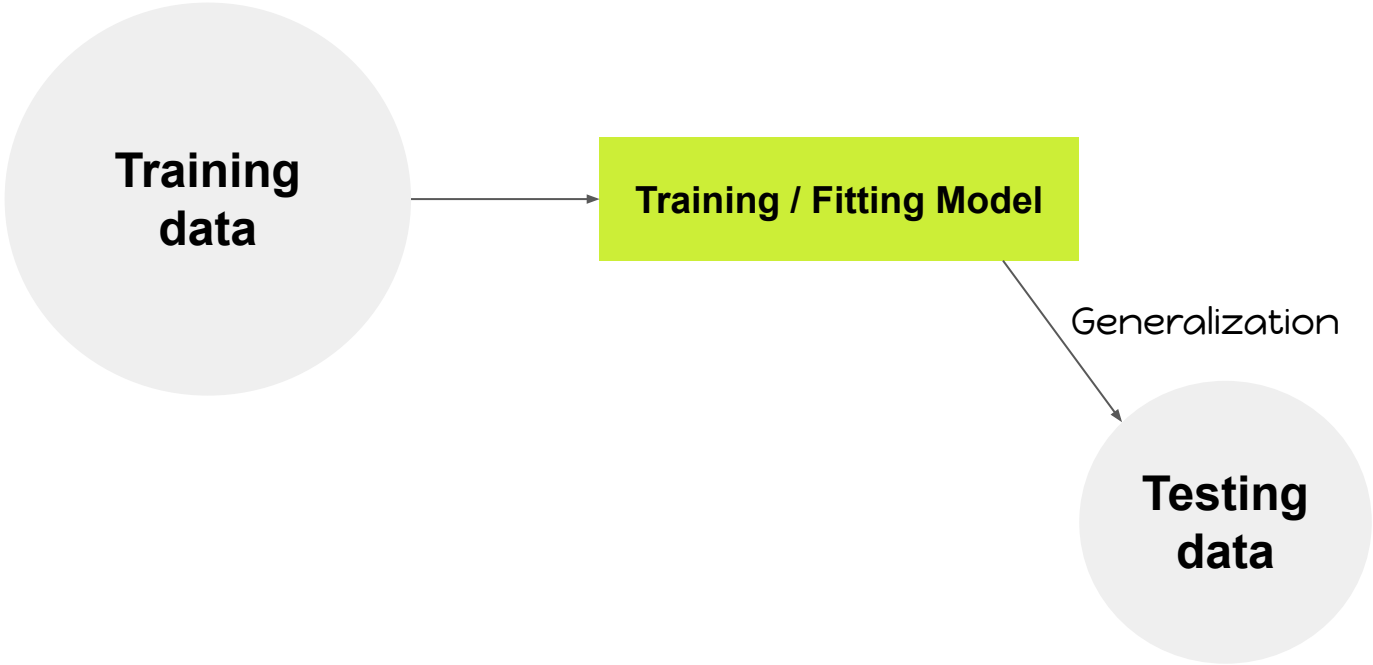
- prepare data
- train algorithm
- test/ evaluate algorithm

ML glossary #2

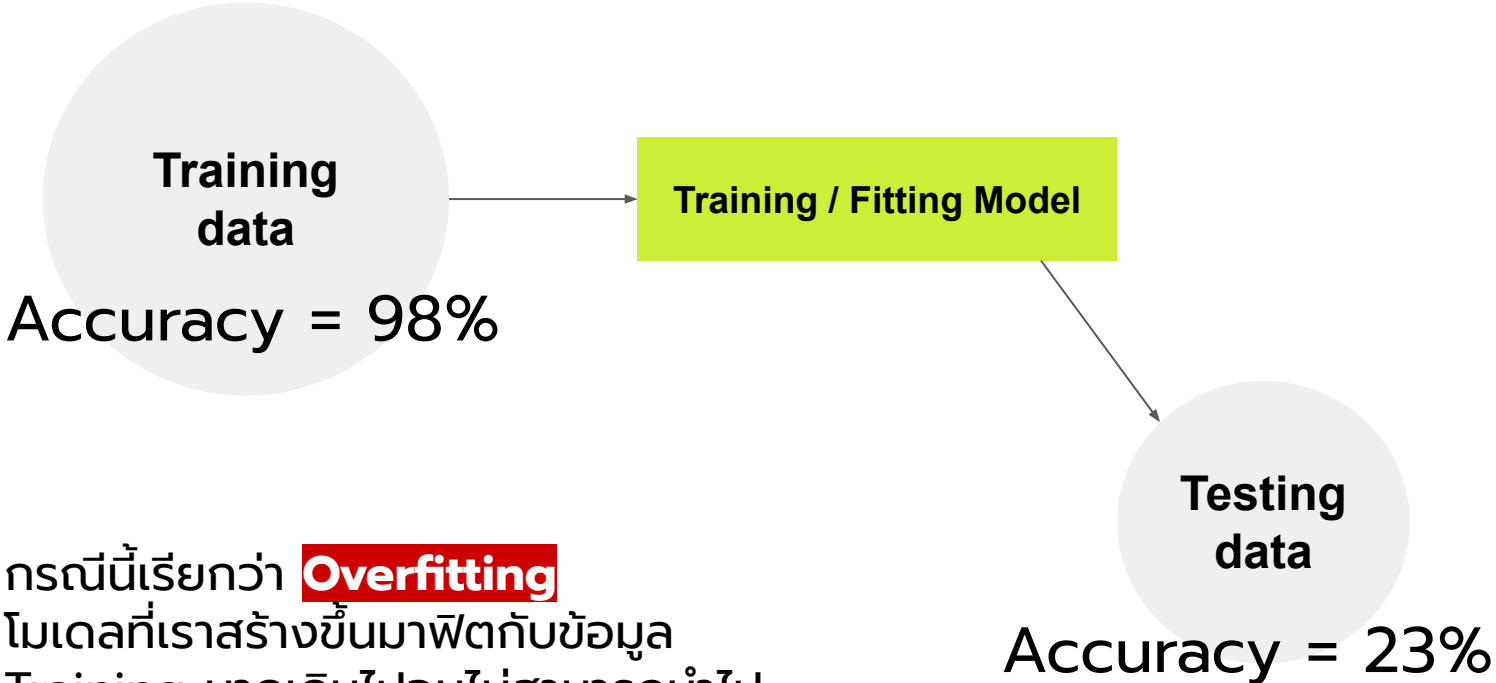
- train test split
- training set
- testing/ validation set
- overfitting



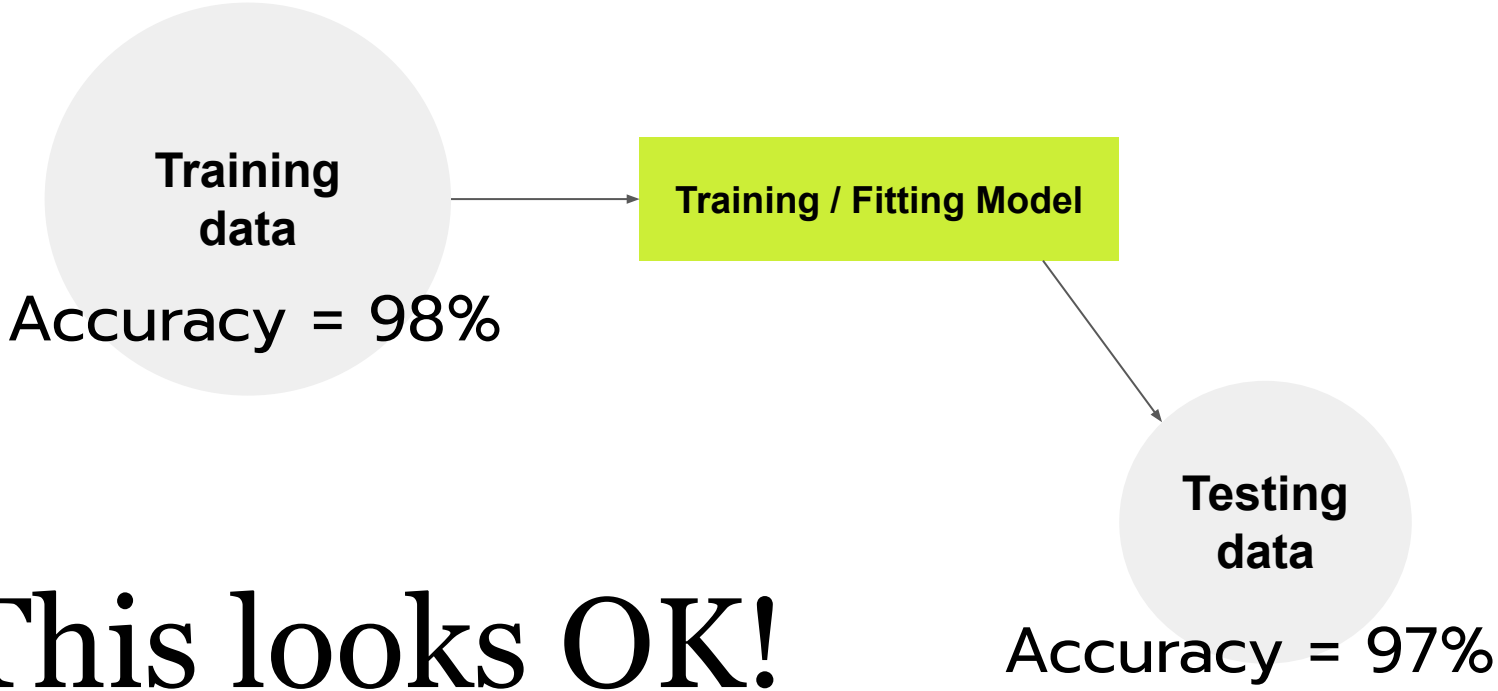
Train test split



เราต้องถามคำถามนี้เสมอ
โมเดลที่เราสร้างขึ้นมาเอาไปใช้จริงได้หรือเปล่า? i.e.
ความถูกต้องของโมเดลกับ test data เป็นเท่าไร



กรณีนี้เรียกว่า **Overfitting**
โมเดลที่เราสร้างขึ้นมาฟิตกับข้อมูล Training มากเกินไปจนไม่สามารถนำไปใช้กับ Testing/ Unseen data ได้



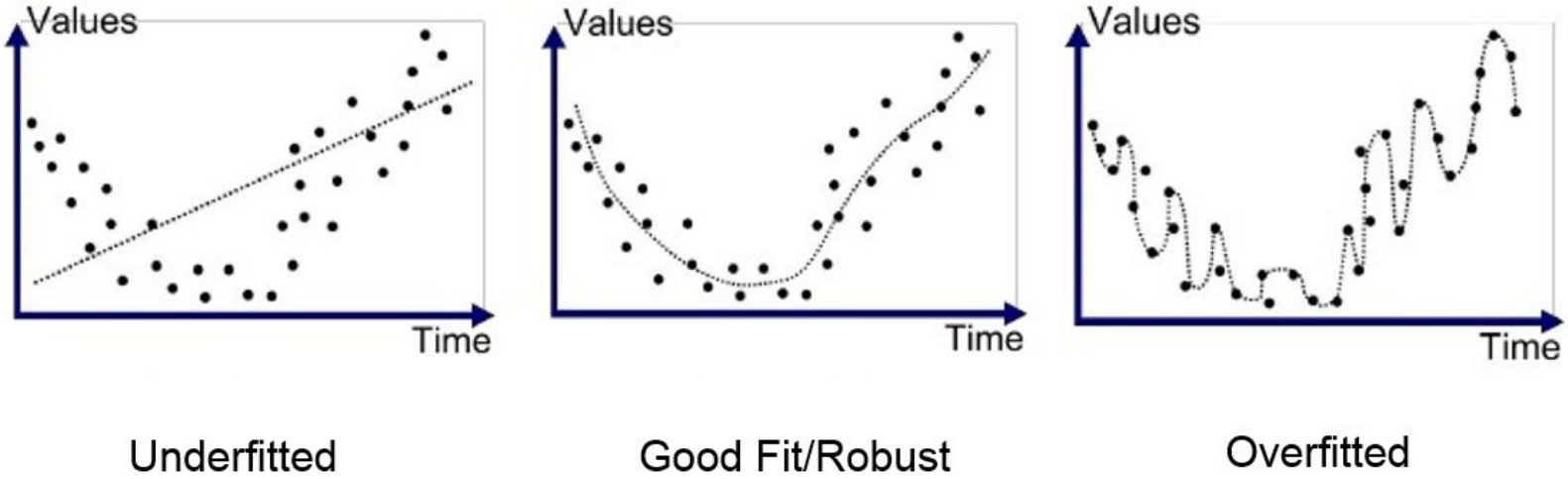
This looks OK!

Rule You Cannot Break

เราจะไม่ทดสอบโมเดลด้วยข้อมูลชุดเดิมที่ใช้เทรนโมเดล

i.e. เราจะไม่ใช้ training data วัดผลว่าโมเดลของเราทำงานดีไหม? แต่ต้องเป็น unseen data ที่โมเดลไม่เคยเห็นมาก่อน

Our goal is in the middle



https://medium.com/greyatom/what-is-underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6803a989c76

Discuss: Overfitting คืออะไร?

เขียนคำตอบได้ที่นี่

Discuss: แล้วถ้า Underfitting ล่ะ?

เขียนคำตอบได้ที่นี่

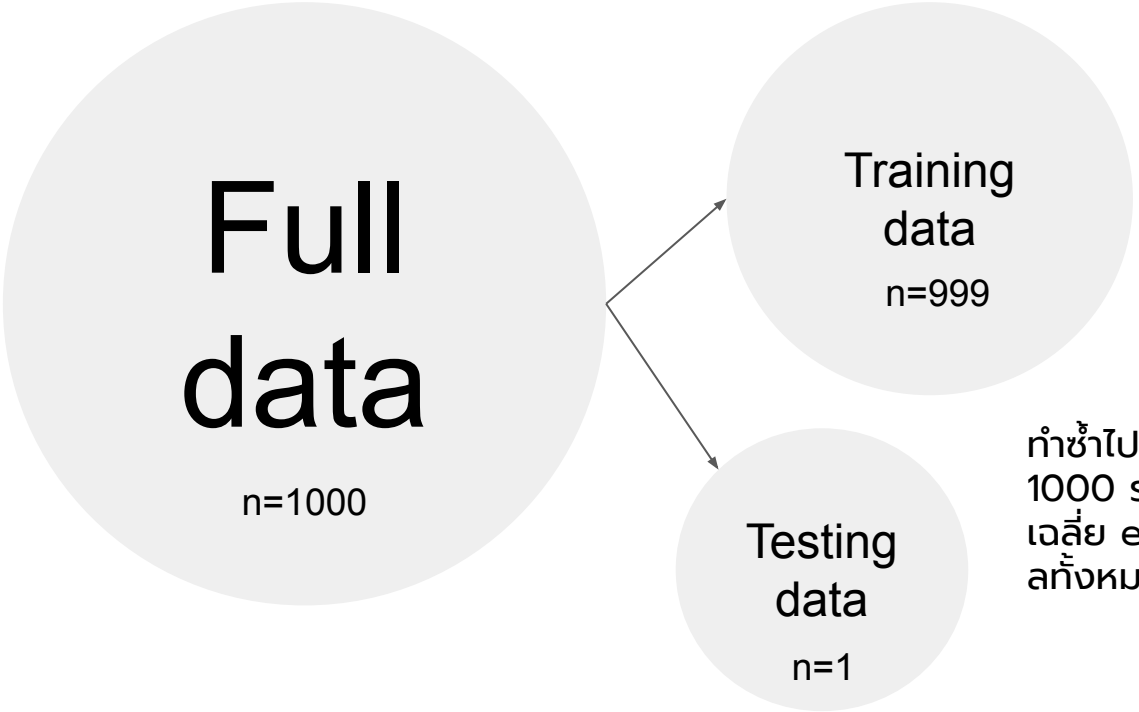
ในทางปฏิบัติ Train Test Split (ส่วนมาก) จะไม่ใช่วิธีที่ดีที่สุดในการสร้างโมเดล ML

เราใช้เทคนิคที่เรียกว่า **Resampling** สำหรับเทรนโมเดลเพื่อผลลัพธ์ที่ดีกว่า

ML glossary #3

- resampling
  - leave one out CV
  - bootstrap
  - k-fold cross validation

Leave One Out CV



ทำซ้ำไปเรื่อยๆจนกว่าจะเทรนโมเดลครบ 1000 รอบ (ตามจำนวน n) แล้วหาค่าเฉลี่ย error หรือ accuracy ของโมเดลทั้งหมด

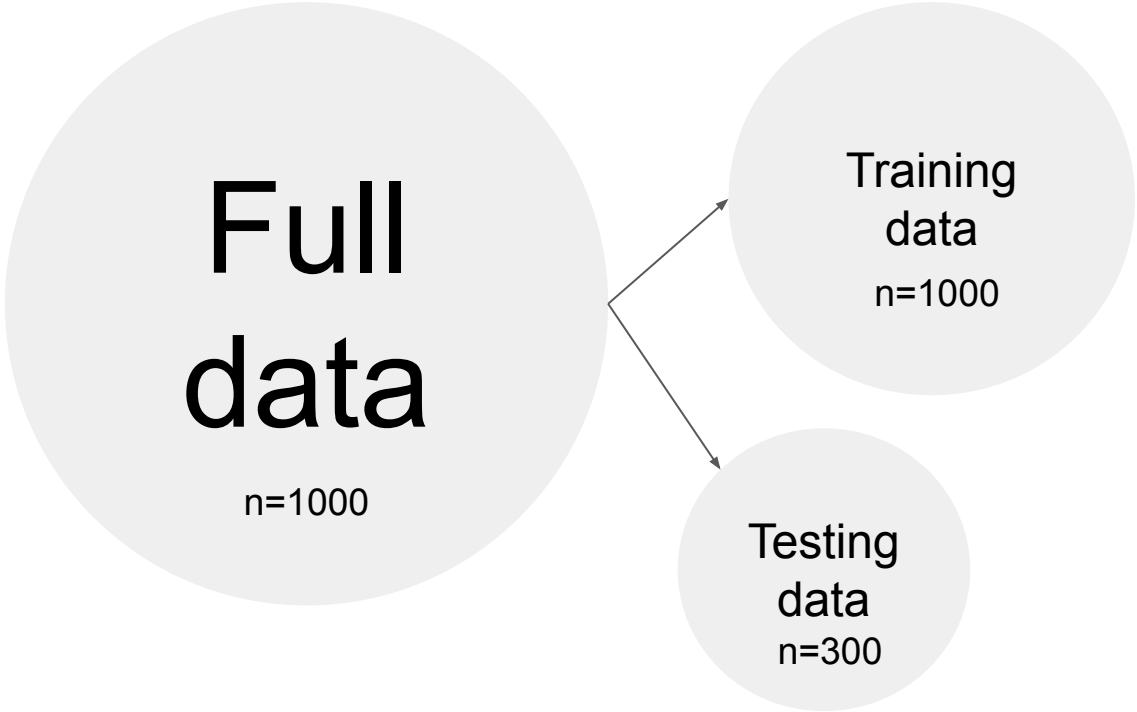
Leave One Out CV

1	2	3	4	...	...	997	998	999	1000	iteration 1
1	2	3	4	...	...	997	998	999	1000	iteration 2
1	2	3	4	...	...	997	998	999	1000	iteration 3
1	2	3	4	...	...	997	998	999	1000	iteration 4
1	2	3	4	...	...	997	998	999	1000	iteration 5

▪ ▪ ▪

1	2	3	4	...	...	997	998	999	1000	iteration 999
1	2	3	4	...	...	997	998	999	1000	iteration 1000

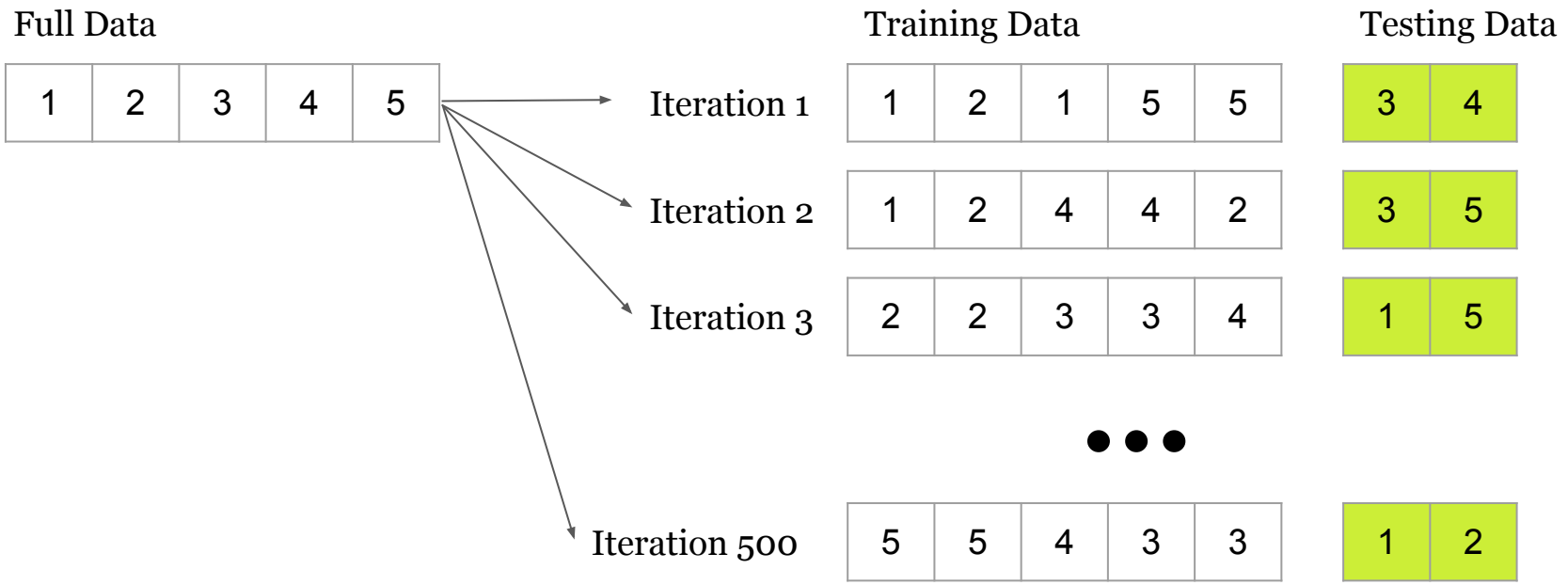
Bootstrap



Sampling with replacement

ใช้การสุ่มซ้ำ n=1000 เหมือน full dataset

Bootstrap



The error will be averaged over 500 training iterations

K-Fold Cross Validation

1	2	3	4	5
1	2	3	4	5
1	2	3	4	5
1	2	3	4	5
1	2	3	4	5

iteration 1: train {2,3,4,5} test {1} -> error 18%

iteration 2: train {1,3,4,5} test {2} -> error 20%

iteration 3: train {1,2,4,5} test {3} -> error 30%

iteration 4: train {1,2,3,5} test {4} -> error 15%

iteration 5: train {1,2,3,4} test {5} -> error 19%

Average error = (18+20+30+15+19) / 5 = 20.4%

ปกติเรานิยมใช้ค่า **K=5** หรือ **K=10**

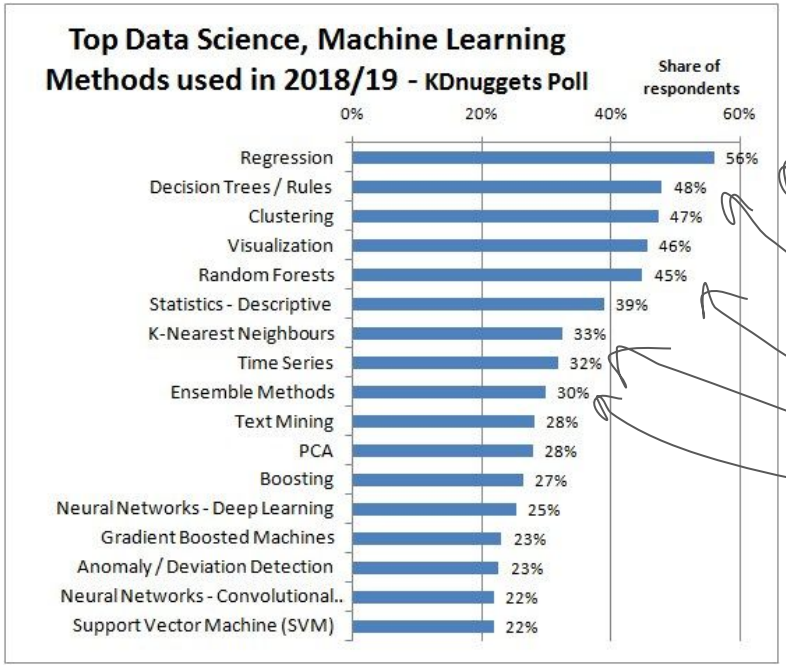
Discuss: LOOCV, Bootstrap, K-Fold ทั้งสามวิธีแตกต่างกันอย่างไร?

เขียนคำตอบได้ที่นี่

Essential ML

- OK what exactly is machine learning
- OK supervised vs. unsupervised
- OK regression vs. classification
- OK train test split vs. cross validation
- model selection + hyperparameter
- model evaluation

Algorithms Ranking



Today we'll cover

- Linear Regression
- Logistic Regression
- Regularized Regression
- Decision Tree
- Random Forests (Bagging)
- K-Nearest Neighbours
- Ensemble Model

https://www.kdnuggets.com/2019/04/top-data-science-machine-learning-methods-2018-2019.html

No Free Lunch

No Free Lunch แปลว่า “ไม่มีโมเดลไหนเก่งที่สุด และสามารถตอบโจทย์ได้ทุกปัญหา”

ถ้ามีใครถามว่าโมเดลไหนเก่งที่สุด?
ให้ตอบว่า “It depends” (ขึ้นอยู่กับข้อมูล)
ความท้าทายของ ML คือการหาโมเดลที่ดีที่สุด
สำหรับปัญหาที่เรากำลังแก้

Occam’s Razor

Algorithm #1 vs. Algorithm #2

ถ้ามีโมเดลสองตัวที่มี performance ดีเท่าๆกัน ให้เลือกตัวที่สร้างและอธิบายได้ง่ายกว่า (**choose simpler model**)

So how can I choose a model?

ให้ลองถาม 2 คำถามง่ายๆนี้

1. ปัญหานี้เป็น regression หรือ classification?
2. อยากได้ high accuracy หรือ high interpretability?

- Always choose a simpler model if performances are similar
- Try different algorithms and find the right one.

caret package

Learn more at https://topepo.github.io/caret/index.html

Max Kuhn

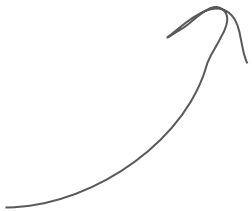
the author of caret package

Dataset for This Tutorial

```
## load library
## install.packages("mlbench")
library(mlbench)
library(tidyverse)

## load dataset for regression
data("BostonHousing")
glimpse(BostonHousing)

## load dataset for classification
data("PimaIndiansDiabetes")
glimpse(PimaIndiansDiabetes)
```

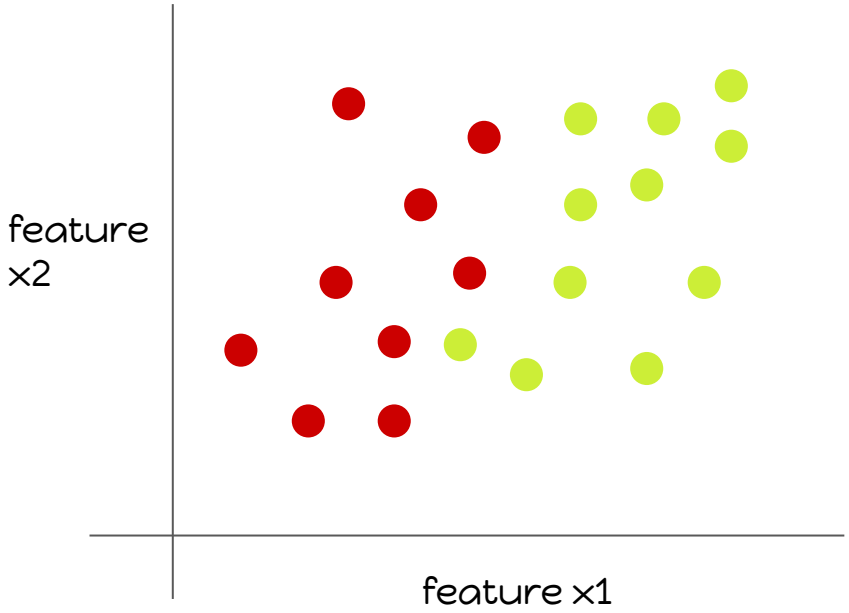
Caret Training Template

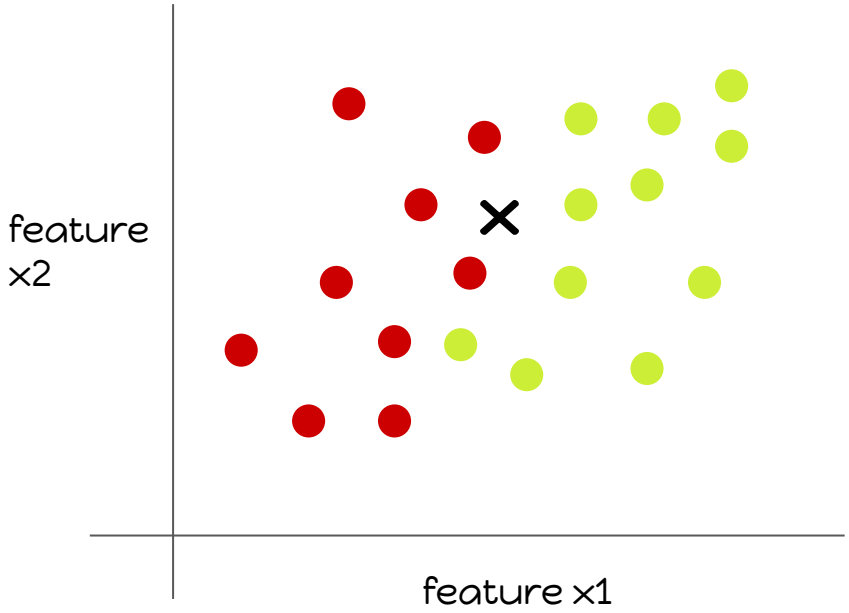
```
model <- train(form = y ~ . ,
               data = train_data ,
               method = “lm” )
```



Model that we want to train

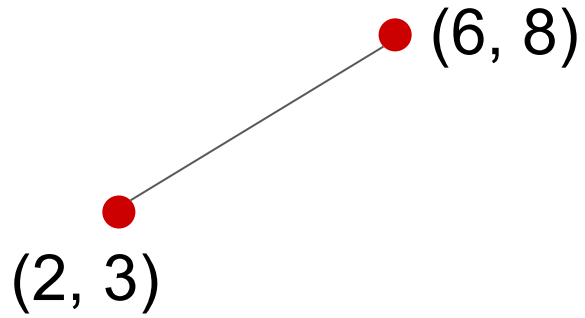
The First Machine, Easiest One to Learn





Euclidean Distance

$$d = \sqrt{(x1 - x2)^2 + (y1 - y2)^2}$$

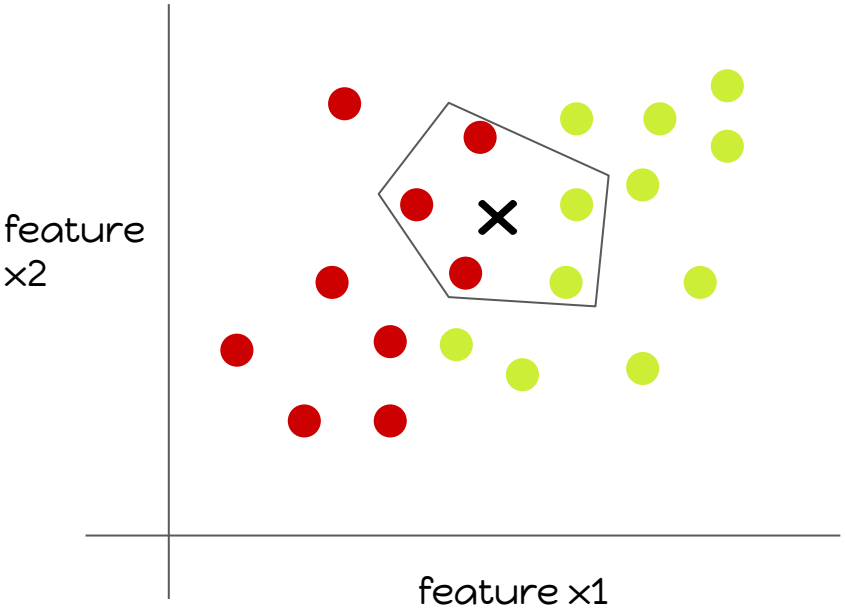


Euclidean Distance

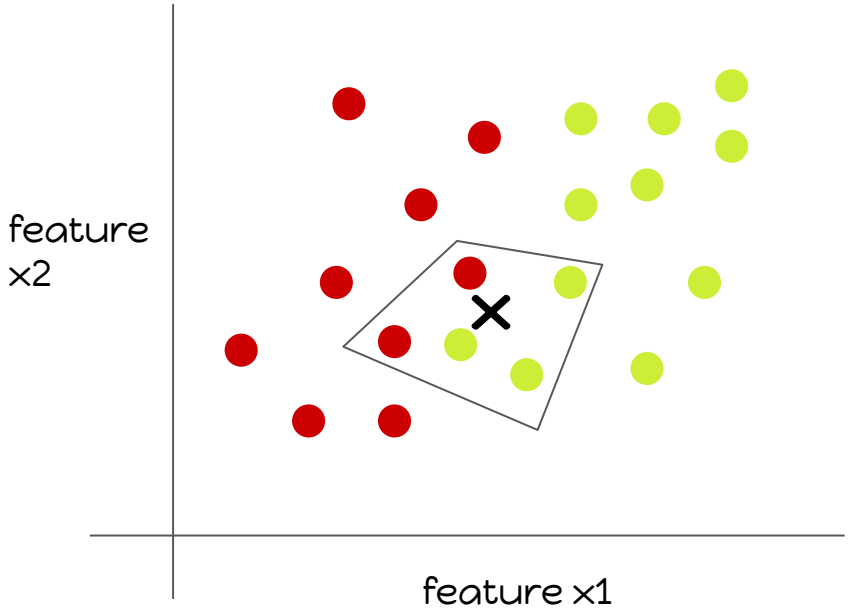
$$d = \sqrt{(x1 - x2)^2 + (y1 - y2)^2}$$

```
point_1 <- c(2,3)
point_2 <- c(6,8)
d <- sqrt( (2-6)**2 + (3-8)**2 )
print(d)
```

We use majority vote to assign label



Predict `Red` 3/5 = 60%



Predict `Green` 3/5 = 60%

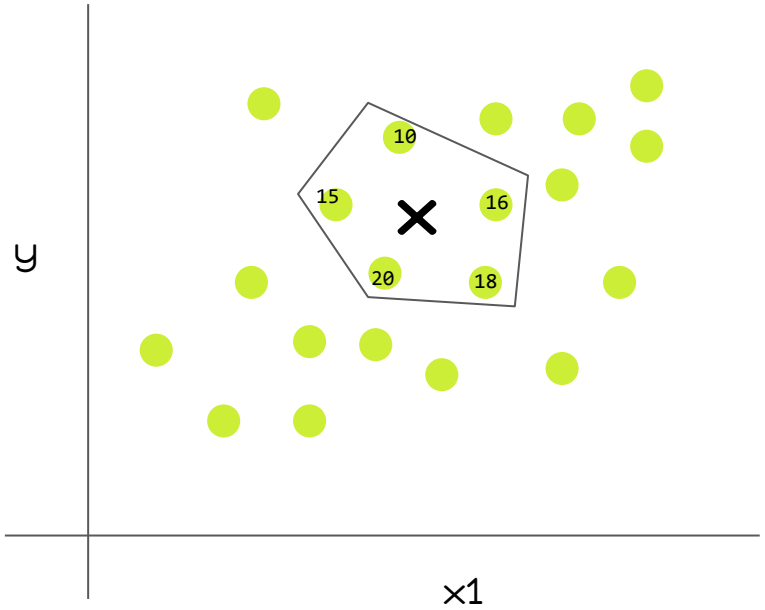
Majority Vote

Steps to train this algorithm

1. แต่งตั้ง สว 250 คน
2. ช่วยกันเลือกนายกๆ

เฮ้ย เด๋วๆๆๆ 55555555555+

We use average vote for regression



Use the average as prediction

(10 + 16 + 18 + 20 + 15) / 5 = 15.8

K-Nearest Neighbors

1. Choose K
2. Compute distance
3. Majority vote for classification or
   Average for regression

Before we train model

Prepare dataset first
We'll use split data into training 75% and testing 25%

```
## split data
set.seed(99)
n <- nrow(BostonHousing)
id <- sample(n, size = n*0.75, replace=FALSE)
train_data <- BostonHousing[id, ]
test_data <- BostonHousing[-id, ]
```

KNN - why so easy!

```
## train model
set.seed(99)
knn_model <- train(medv ~ .,
                   data = train_data,
                   method = "knn")

## test model
p <- predict(knn_model, newdata = test_data)

## rmse
rmse <- sqrt(mean((p - test_data$medv)**2))
```

KNN + K-Fold CV

```
## train model
set.seed(99)
ctrl <- trainControl(method = “cv”, number = 5, verboseIter = TRUE)
knn_model <- train(medv ~ .,
                   data = train_data,
                   method = "knn",
                   trControl = ctrl)

## test model
p <- predict(knn_model, newdata = test_data)

## rmse
rmse <- sqrt(mean((p - test_data$medv)**2))
```

Random Search

```
## train model
set.seed(99)
ctrl <- trainControl(method = “cv”, number = 5, verboseIter = TRUE)
knn_model <- train(medv ~ .,
                   data = train_data,
                   tuneLength = 5,
                   method = "knn",
                   trControl = ctrl)

## test model
p <- predict(knn_model, newdata = test_data)

## rmse
rmse <- sqrt(mean((p - test_data$medv)**2))
```

Try 5 values of K

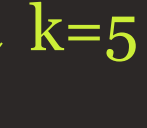
Grid Search

```
## create grid
myGrid <- expand.grid(k = 1:10)

## train model
set.seed(99)
ctrl <- trainControl(method = “cv”, number = 5, verboseIter = TRUE)
knn_model <- train(medv ~ .,
                   data = train_data,
                   tuneGrid = myGrid,
                   method = "knn",
                   trControl = ctrl)

## test model
p <- predict(knn_model, newdata = test_data)

## rmse
rmse <- sqrt(mean((p - test_data$medv)**2))
```

The values we select ourselve :D

Grid Search Result

```
> knn_model
k-Nearest Neighbors

379 samples
 13 predictor

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 303, 303, 303, 303, 304
Resampling results across tuning parameters:

  k   RMSE      Rsquared   MAE
   1  7.607647  0.4598462  4.922035
   2  6.857741  0.5208036  4.750591
   3  6.778822  0.5139722  4.657967
   4  6.659557  0.5153593  4.651180
   5  6.646851  0.5131619  4.681624
   6  6.660705  0.5081547  4.648119
   7  6.711653  0.5001402  4.661983
   8  6.881981  0.4749499  4.749852
   9  6.872293  0.4768345  4.763948
  10  6.927021  0.4683690  4.773996

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

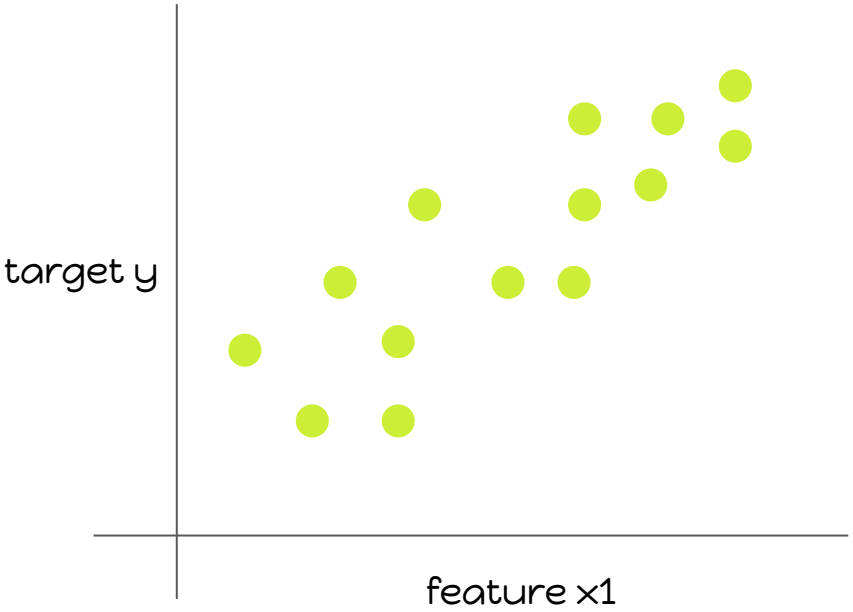


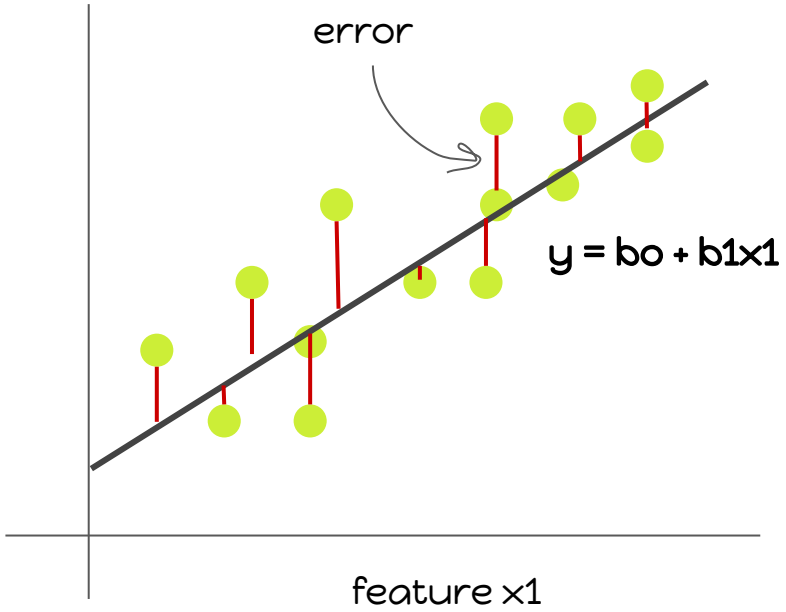
Cross Validation ช่วยเราเลือกค่า k ที่ทำให้ RMSE ต่ำที่สุด ตอนเรา train model

Key Learning

1. KNN เข้าใจง่ายทำงานได้โอเคร ถ้า feature ไม่เยอะมาก
2. KNN ใช้ได้ทั้ง regression/ classification
3. K ใน KNN คือค่า hyperparameter ที่เราเปลี่ยนได้
4. เราเลือก K ที่ทำให้ train RMSE ต่ำที่สุด
5. train RMSE ต่ำที่สุดไม่ได้แปลว่าโมเดลเราจะทำนาย test_data ได้ดี ต้องเอาไปทดสอบอีกที

Linear Regression Explained





Linear Regression

$$\hat{y} = B_0 + B_1 X_1$$

prediction

y intercept

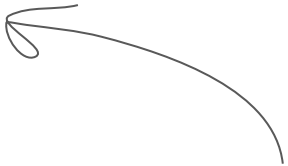
slope

Linear Regression finds B0 and B1 that minimize the error

Linear Reg. aims to minimize the sum of squared error

$$minimize \sum (prediction - actual)^2$$

$$minimize \sum (\hat{y} - y)^2$$



Sum of Squared Error or RSS (for short)

Few Regression Metrics You Should Know

$$MAE = \frac{1}{n} * \sum |\hat{y} - y|$$

$$MSE = \frac{1}{n} * \sum (\hat{y} - y)^2$$

$$RMSE = \sqrt{\frac{1}{n} * \sum (\hat{y} - y)^2}$$

โมเดลที่เราเทรนจะพยายามทำให้ค่า MAE/ MSE/ RMSE มีค่าต่ำที่สุด i.e. minimize error

It’s very easy to compute

y	y_hat	error	|error|	error^2	
10	8.5	1.5	1.5	2.25	
12	14.5	-2.5	2.5	6.25	
14	10	4	4	16	
16	17	-1	1	1	
18	17.5	0.5	0.5	0.25	
			9.5	**25.75**	
			1.9	5.2	2.3
			MAE	MSE	RMSE

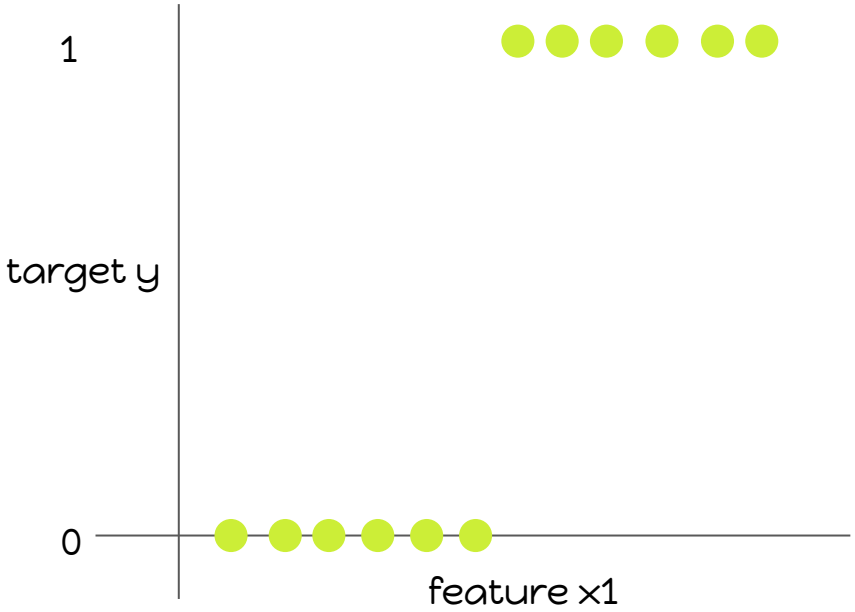
Linear Regression

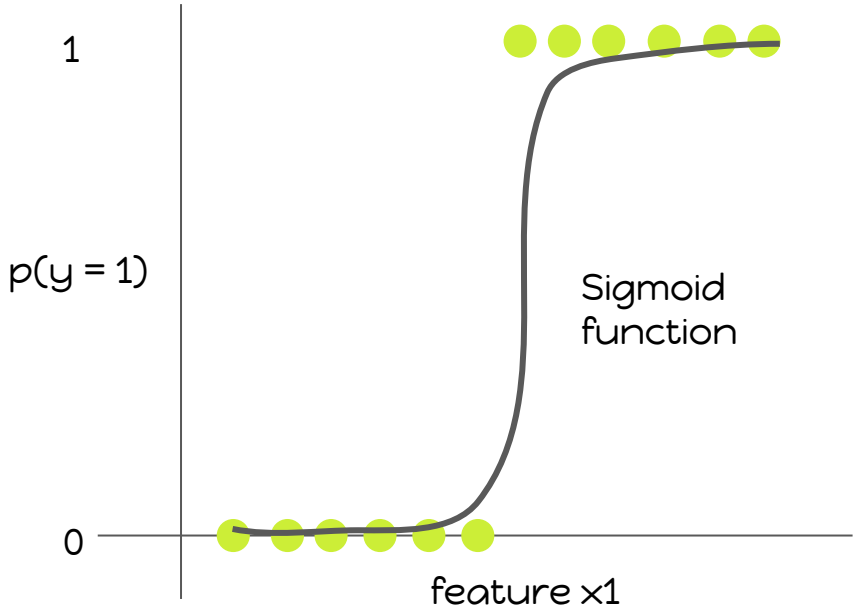
```
## train model with train_data
set.seed(99)
lm_model <- train(medv ~ rm + indus + crim,
                  data = train_data,
                  method = "lm")

## test model (predict test data)
p <- predict(lm_model, newdata = test_data)
rmse <- sqrt(mean( (p - test_data$medv)** 2 ))
```

Linear Regression + K-Fold CV

```
## train model with train_data
set.seed(99)
ctrl <- trainControl(method = “cv”, number = 5,
                     verboseIter = TRUE)

lm_model <- train(medv ~ rm + indus + crim,
                  data = train_data,
                  method = "lm",
                  trControl = ctrl)

## test model (predict test data)
p <- predict(lm_model, newdata = test_data)
rmse <- sqrt(mean( (p - test_data$medv)** 2 ))
```

Logistic Regression Explained





Logistic Regression is an Extended version of Linear Regression

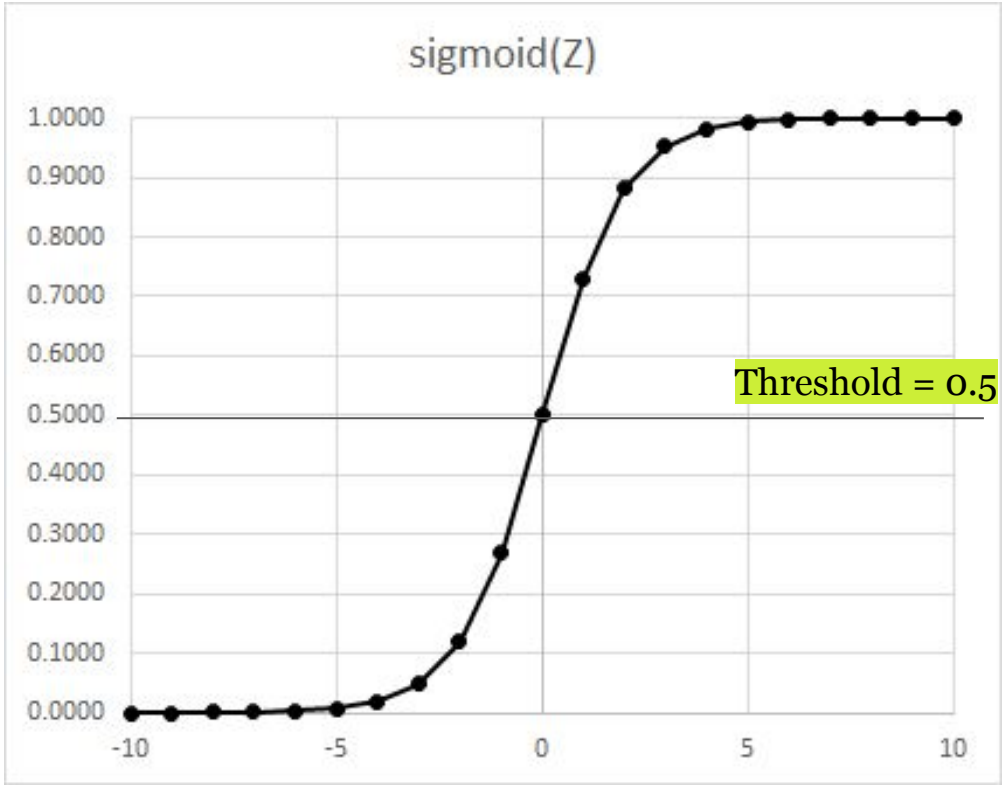
$$Z = B_0 + B_1 X_1$$

$$P(Y = 1|x) = \frac{e^z}{1 + e^z}$$

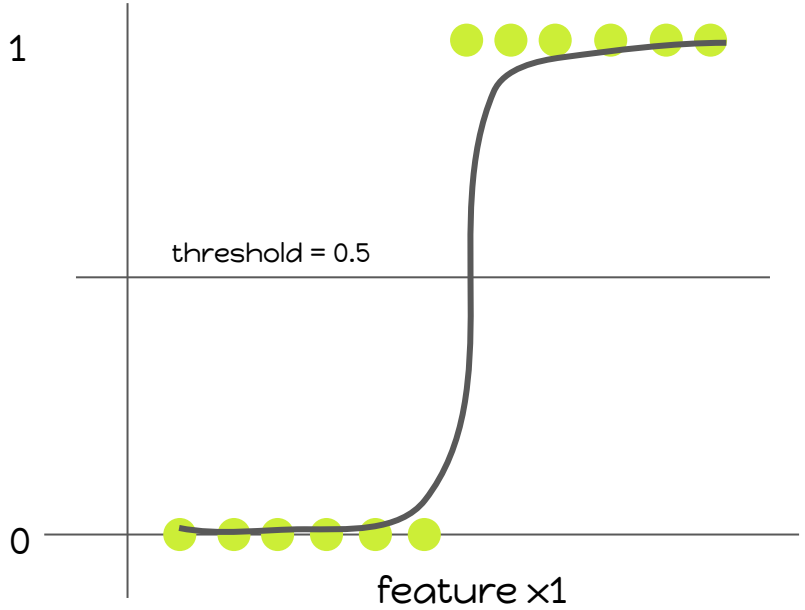
Sigmoid function

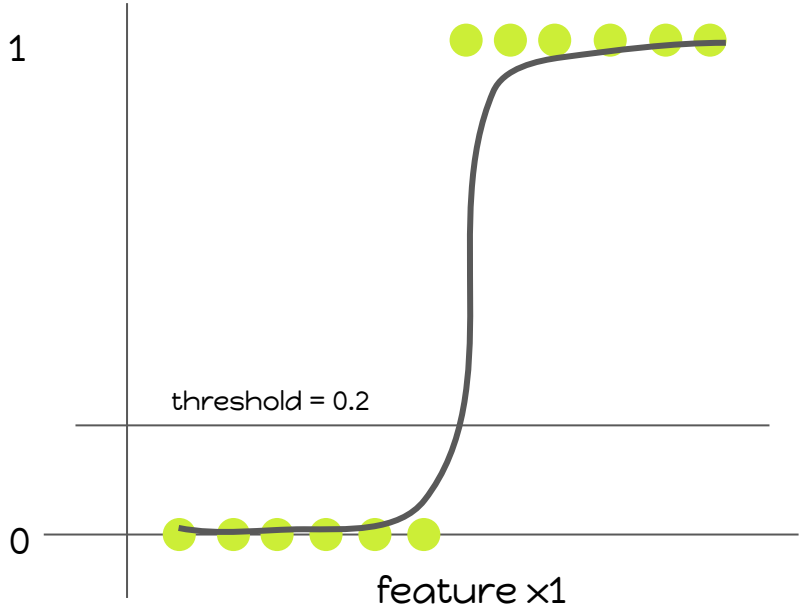
Z	sigmoid(Z)	y_hat
-10	0.0000	0
-9	0.0001	0
-8	0.0003	0
-7	0.0009	0
-6	0.0025	0
-5	0.0067	0
-4	0.0180	0
-3	0.0474	0
-2	0.1192	0
-1	0.2689	0
0	0.5000	0
1	0.7311	1
2	0.8808	1
3	0.9526	1
4	0.9820	1
5	0.9933	1
6	0.9975	1
7	0.9991	1
8	0.9997	1
9	0.9999	1
10	1.0000	1

If sigmoid(Z) > 0.5, predict y = 1, else y = 0



Our prediction will change if threshold changed





Logistic Regression + K-Fold CV

```
## train model with train_data
set.seed(99)
ctrl <- trainControl(method = “cv”, number = 5,
                     verboseIter = TRUE)

logistic_model <- train(diabetes ~ .,
                        data = train_data,
                        method = "glm",
                        trControl = ctrl)

## test model (predict test data)
p <- predict(logistic_model, newdata = test_data)
accuracy <- mean(p == test_data$diabetes)
```

Classification Metrics

- Accuracy
- Precision
- Recall
- F1

สามารถคำนวณได้
ง่ายๆจาก
Confusion Matrix

Create Confusion Matrix

```
## use table()
table(predicted, actual, dnn = c("predicted", "actual"))
```

		Actual	
		neg	pos
Predicted	neg	101	33
	pos	14	44

```
## how we calculate four metrics
accuracy <- (101 + 44) / (101 + 33 + 44 + 14)
precision <- 44 / (44 + 14)
recall <- 44 / (44 + 33)
F1 <- 2 * (precision * recall) / (precision + recall)
```

Meaning of the four metrics

Metrics	**ความหมาย**
Accuracy	ความถูกต้องของโมเดลในภาพรวม
Precision	ทุก 100 ครั้งที่เราทำนาย y=1 โอกาสถูกเท่าไร
Recall	ทุกผู้ป่วยจริงๆ 100 คน เราตรวจเจอกี่คน
F1	ค่าเฉลี่ยระหว่าง precision, recall

Regularized Regression

1. Ridge Regression
2. Lasso Regression

Regularization is a key technique in ML to reduce overfitting :D

Ridge Regression (L2)

$$RSS = \sum (\hat{y} - y)^2$$

Normal RSS from Linear Regression

$$Ridge\ RSS = \sum (\hat{y} - y)^2 + \lambda \sum \beta^2$$

Ridge add this term to the error function

Lasso Regression (L1)

$$RSS = \sum (\hat{y} - y)^2$$

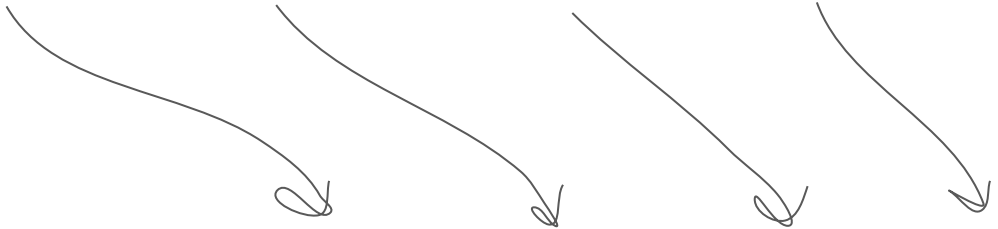
Normal RSS from Linear Regression

$$Lasso\ RSS = \sum (\hat{y} - y)^2 + \lambda \sum |\beta|$$

Lasso add this term to the error function

Why Ridge/ Lasso works well in practice?

```
y_hat  = bo + b1x1 + b2x2 + b3x3 + b4x4
y_hat  = 100 + 150x1 + 200x2 + 120x3 + 80x4
```



$$\text{Lasso} = \text{RSS} + \text{Lambda} * (150 + 200 + 120 + 80)$$

Lambda = 0
Error is the same as Linear Regression

Lambda > 0
All coefficient (B) in the model must be shrunken to reduce the new error

ElasticNet + K-Fold CV

```
## train elasticnet model
set.seed(99)
ctrl <- trainControl(method = “cv”, number = 5,
                     verboseIter = TRUE)

enet_model <- train(diabetes ~ .,
                    data = train_data,
                    method = "glmnet",
                    trControl = ctrl)

## test model
p <- predict(enet_model, newdata = test_data)
accuracy <- mean(p == test_data$diabetes)
```

ElasticNet = Mixed between Ridge + Lasso

KNN + 5-Fold CV + Grid Search K=1:10

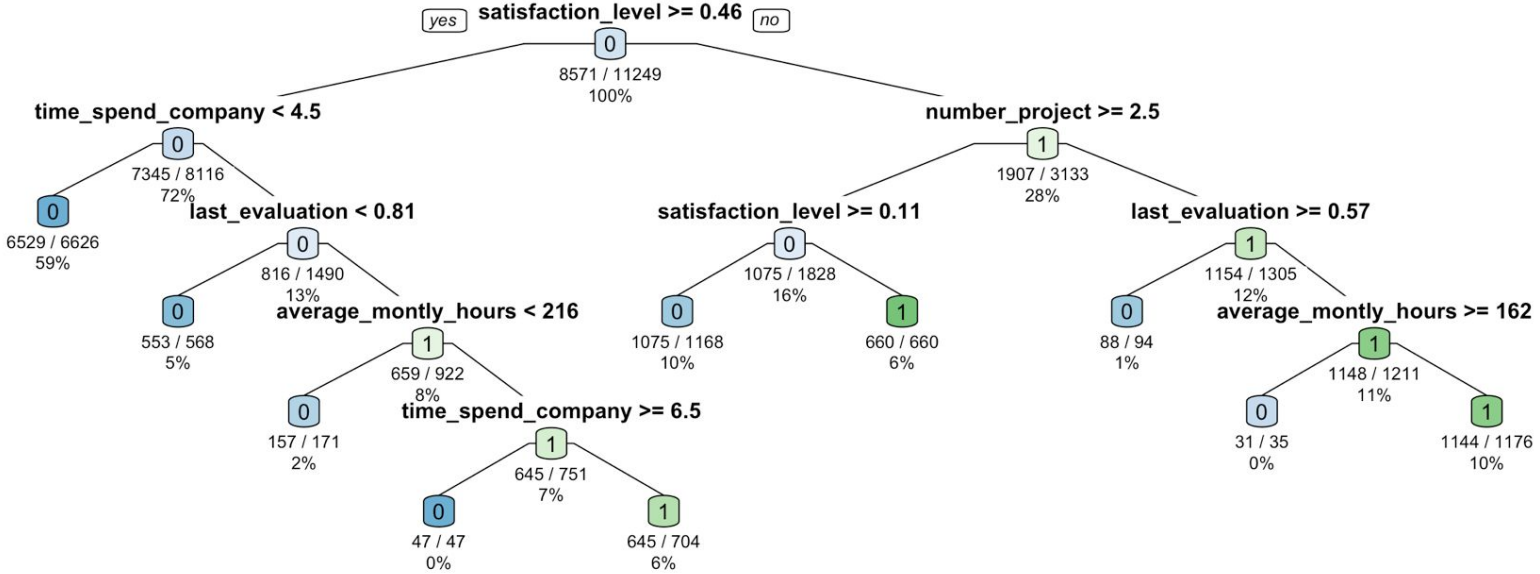
```
## train KNN
set.seed(99)
myGrid <- expand.grid(k = ....)
ctrl <- trainControl(method = "....", number = ....,
                     verboseIter = TRUE)

knn_model <- train(diabetes ~ .,
                           data = ....,
                           method = "....",
                           tuneGrid = ....
                           trControl = ....)

## test model
p <- predict(enet_model, newdata = ....)
accuracy <- mean(....)
```

Which Model is Better?

Time for A Fun GAME :D

- Ask me a yes/no question
- To guess my favourite animal
- Max 10 questions

Decision Tree



How Decision Tree Work?

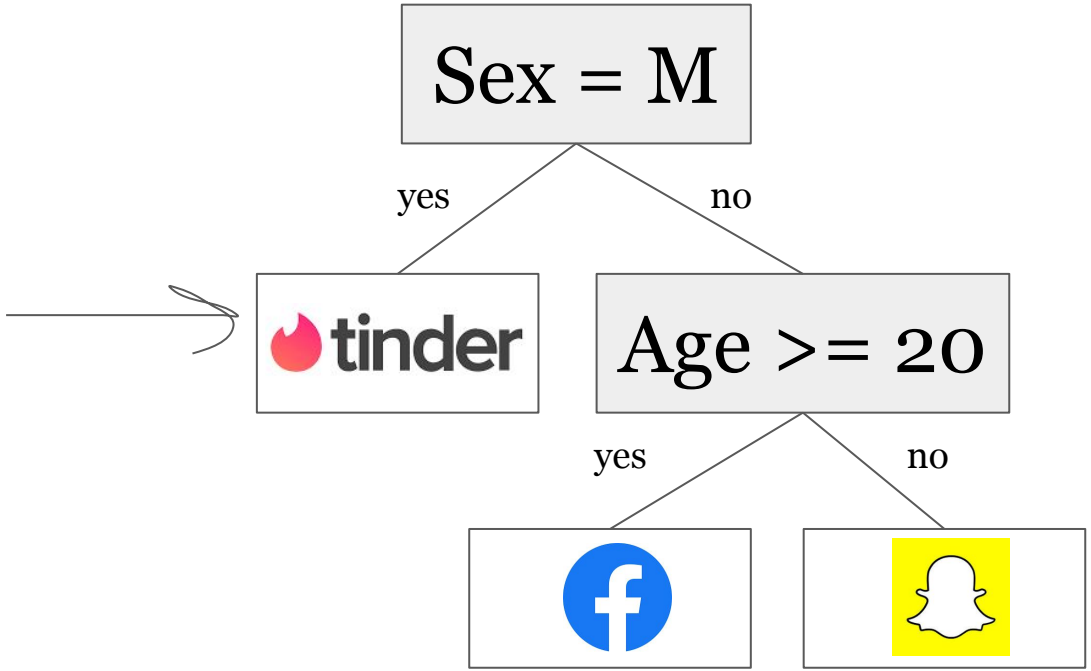
Age	Sex	App
15	M	tinder
20	M	tinder
16	F	
19	M	tinder
22	F	
20	F	

เวลาเราสร้าง decision tree เราถามคำถาม yes/no question ทีละข้อ

i.e. feature ใช้แบ่ง App ได้ดีที่สุด

How Decision Tree Work?

Age	Sex	App
15	M	tinder
20	M	tinder
16	F	
19	M	tinder
22	F	
20	F	



Decision Tree + 10-Fold CV

```
## train tree
set.seed(99)

ctrl <- trainControl(method = "....", number = ....,
                     verboseIter = TRUE)

tree_model <- train(diabetes ~ .,
                         data = ....,
                         method = "rpart",
                         trControl = ....)

## test model
p <- predict(tree_model, newdata = ....)
accuracy <- mean(....)
```

Let’s do a quick review

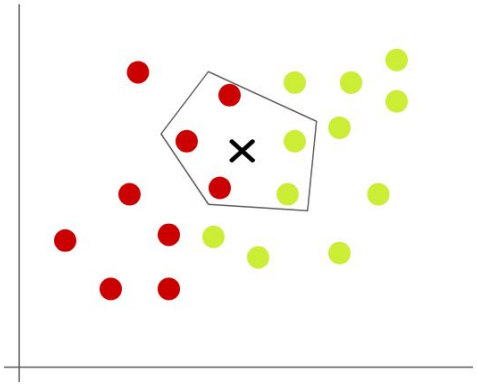
Parametric	**Non-Parametric**
Linear Regression	KNN
Logistic Regression	Decision Tree
Ridge Regression	Random Forest
Lasso Regression	

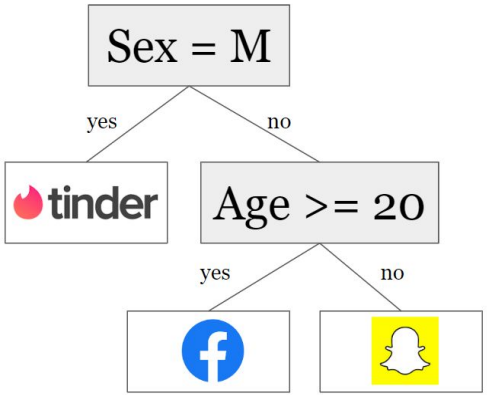
Regression is a Linear Combination (Parametric)

$$y\_hat = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + b_4x_4$$

มีฟอร์ม

While a non-parametric has no form :P





Random Forest

We grow hundreds of uncorrelated (decision) trees

Combine them to make prediction (similar to KNN, majority vote or average)

What's uncorrelated trees?

Bootstrap + mtry hyperparameter

All features

X1	X2	X3	X4	X5

X1	X3	X4

Tree 1

X2	X3	X5

Tree 2

X1	X2	X5

Tree 3

X2	X3	X4

Tree 100

Random Forests + 5-Fold CV + Grid Search

```
## train random forests
set.seed(99)
myGrid <- expand.grid(mtry = 2:4)

ctrl <- trainControl(method = "....", number = ....,
                     verboseIter = TRUE)

rf_model <- train(diabetes ~ .,
                          data = ....,
                          method = "rf",
                           tuneGrid = myGrid,
                          trControl = ....)

## test model
p <- predict(rf_model, newdata = ....)
accuracy <- mean(....)
```

Ensemble

Ensemble

American pronunciation

aan · saam · bl

Slow

Feedback

อาน ซาม เบิ้ล

Ensemble

นำโมเดลหลายๆตัวมาช่วยกันทำนายผล (Majority Vote)

KNN	Logistic Regression	Ridge Regression	Decision Tree	Random Forest
1	0	0	1	1

Save Trained Model

```
saveRDS(model, “model.rds”)

model <- readRDS(“model.rds”)
```

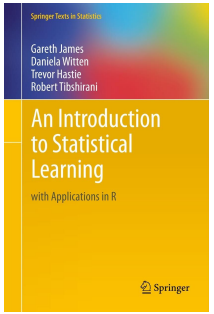
ML Summary

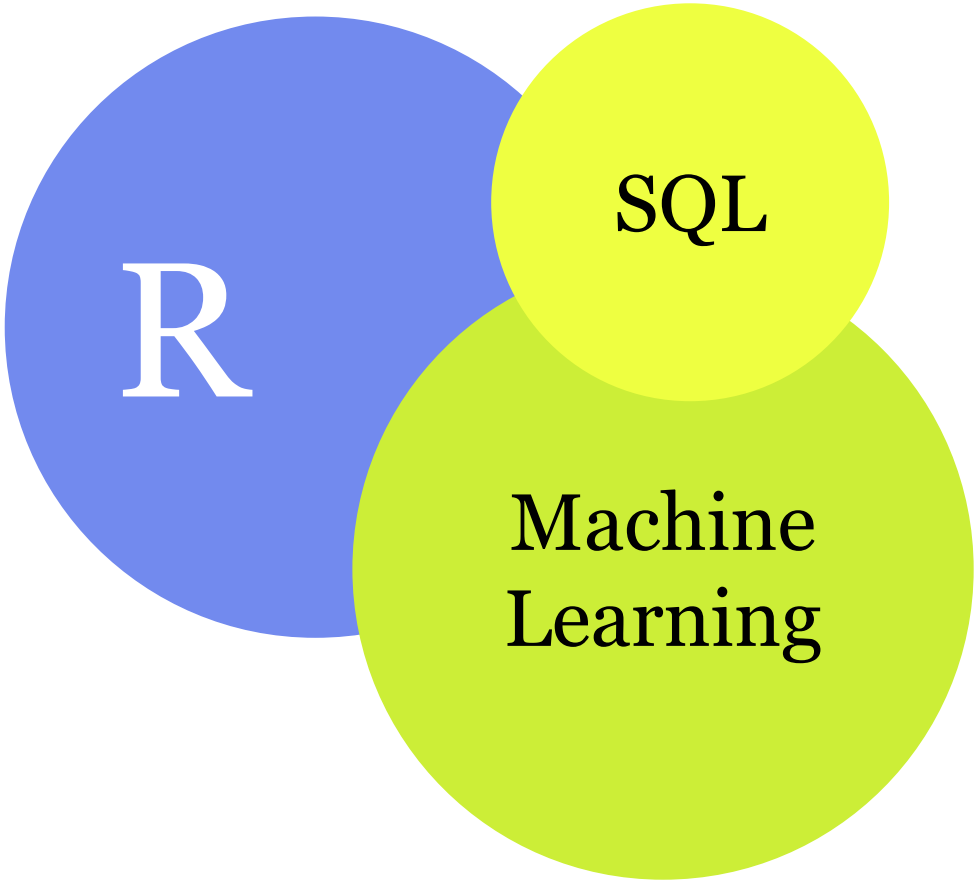
- Machine Learning is **art + science**
- Try different models (No Free Lunch)
- Choose the simpler model
- Use CV + Grid Search to fine-tune model
- Start with Regression or decision tree because they are very fast to train

Learn More

Introduction to Statistical Learning

http://www-bcf.usc.edu/~gareth/ISL/



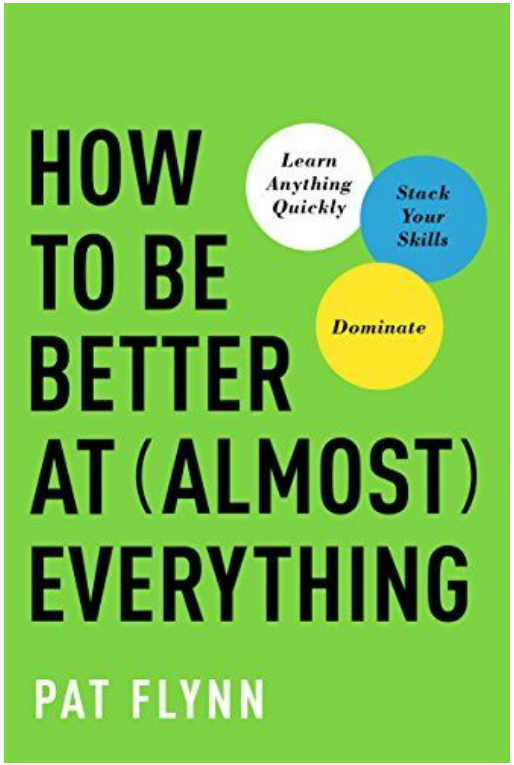

R
SQL
Machine
Learning


Piece of Advice
Learning how to learn

The most important skill is learning

Always learn new thing

Able to find answers on the internet (Google skill)



Breadth of skills > Specialization

(not just data science)

Marketing Skills

Skills that get you paid

Interest Skills

Data Science, Coding, I guess :)

Foundation Skills

English, Discipline, Googling, Logic, Persuasion, Faith


Discipline
equals
Freedom

Read 45 minutes a day

You can finish a book in a week

The job of an educator is to teach students to see **vitality** in themselves

- Joseph Campbell

2-Min Survey

Certificate

สมัครเรียนฟรี คอร์สออนไลน์ของเพจเราบน https://datarockie.blog