



Machine Learning Live Class

☰ Tags	Live Class ML
➤ Class	
☑ Finished Yet?	☑
➤ Knowledge	🍌 <u>Live Classes</u>

ML Intro Notes:

Part 1

Slide + Materials:

-Slide

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/65c4310f-a5c6-4950-8ae1-e23a264d689c/Live_07_-_Intro_to_Machine_Learning.pdf

-Lecture (เป็นคู่มือให้ไปนั่งอ่าน)

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/bdaa654f-daab-436d-bef8-e463f18bcc91/ML_lecture_2-compressed.pdf

-churn.csv:

<https://s3-us-west-2.amazonaws.com/secure.notion-static.com/26349d6b-7f50-48fa-be3e-b6ed6084c4b0/churn.csv>

-bankmarketing_train.csv:

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/a13a3d09-9e8f-4a71-87e0-3e9a59b7c2e0/bankmarketing_train.csv

Simple ML Pipeline

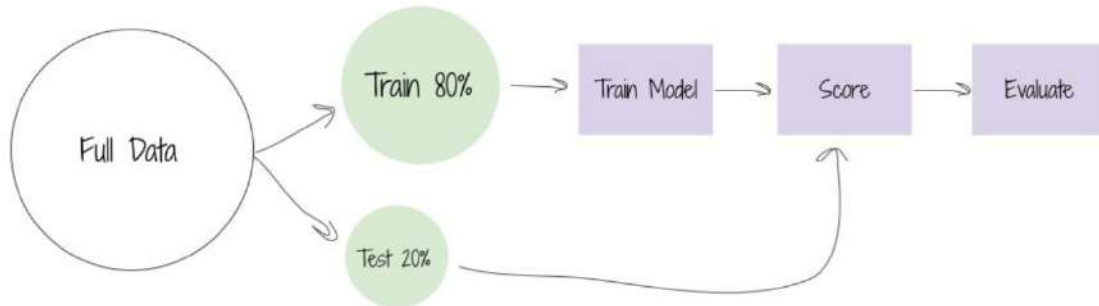
1. Split data / Prepare data
2. Train
3. Score
4. Evaluate

รูปแบบการ Train Model ที่ง่ายที่สุดคือ Train Test Split (แบ่งเป็น 70:30 หรือ 80:20 ก็ได้ ไม่มีกฎตายตัว)

เราสามารถเขียน Function เพื่อ Split data รวมถึงคำนวณค่า Error Metrics ต่าง ๆ ได้ด้วยตัวเอง ซึ่ง Function ของ Regression ที่เราใช้บ่อย ๆ จะมี MAE, MSE, และ RMSE ซึ่ง Metrics พวกนี้เป็น Absolute Metrics ยิ่งค่าเข้าใกล้ศูนย์ แปลว่า Model ของเรากำหนดผลได้ดี ถ้าค่าเป็น 0 คือทายถูกหมดเลย (ซึ่งในทางปฏิบัติแทบจะเป็นไปไม่ได้เลย โมเดลไม่ได้แม่นยำ 100%)

Resampling: เทคนิคมาตรฐานที่เราใช้ Train Model คือ K-Fold Cross Validation โดยทั่วไปเราจะใช้ $K = 5$ หรือ $K = 10$ เพื่อประหยัดเวลาในการ Train Model แต่ได้ผลลัพธ์ที่ใช้งานได้จริง

R Simple pipeline to build ML models



^Train Test Split Pipeline

-เราอยากแบ่งข้อมูลให้ Test Data ใหญ่พอที่จะทดสอบ Model ของเราได้อย่างมีประสิทธิภาพ

-R Squared = Coefficient of Determination (เข้าใกล้ 1 ถือว่าดี เข้าใกล้ 0 ถือว่าไม่ดี)

-RMSE = Actual - Prediction แล้วยกกำลังสอง หาค่าเฉลี่ย แล้วถอดรากที่สองออกมาอีกที (Root Mean Squared Error)

-เราสามารถ Build Model ได้หลายแบบ (ใช้หลาย Algorithm เช่น Linear Regression, Decision Tree, และ Neural Network) แล้วค่อยวัดว่า Model แบบไหนมีประสิทธิภาพดีที่สุด ถึงค่อยเลือกตัวนั้นมาใช้

*เพราะเราไม่รู้เลยว่า Algorithm ไหนจะใช้แก้ไขปัญหได้ดีที่สุด เราสามารถ 'ทดลอง' (Experiment) ใช้ Algorithm หลาย ๆ แบบในการ Train Model ของเราแล้วเลือก Model ที่ได้ผลลัพธ์ที่ดีที่สุดได้ [ML คือการทดลอง]

-Model ที่อธิบายง่าย ประสิทธิภาพอาจไม่มาก แต่ Model ที่อธิบายยาก (Black Box) ประสิทธิภาพอาจดีกว่า

-ML ไม่ได้เหมาะกับทุกปัญหา (เช่นการ Predict ราคาพร้อมกันยอดขาย) *Problem Framing

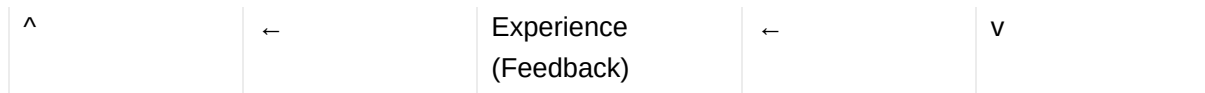
Essential ML

-Machine Learning: Field of study that gives computers the ability to learn without being explicitly programmed

-While human learns from experience, computer learns from data.

-Simple idea:

Computer	→	Task	→	Performance



-Dataset: ข้อมูลทั้งหมด

-Data Point: 1 row ที่อยู่ใน Dataset (Feature ใช้เรียก column)

-Label หรือ Target: column ที่เราใช้เป็นตัวแปรตาม (y)

-Supervised Learning:

Features (X) 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												5.21	28.7
7	0.06629												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	6.0621	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	61.6	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

Supervised Learning

Dataset: BostonHousing

Mapping

มี x มี y หรือ Function (Function Approximation) ใช้ในการทำนาย (Prediction)

-Deterministic Algorithm: ให้ Input (x) แบบเดิม Output (y) ก็จะออกมาแบบเดิมเสมอ

Unsupervised Learning:

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	6.996	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

Dataset: BostonHousing

Unsupervised Learning

ใช้ในการจัดกลุ่ม (Clustering เช่น K-Means Clustering)

ความแตกต่างระหว่าง Supervised และ Unsupervised Learning:

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 <ul style="list-style-type: none"> - Regression - Classification 	Example algorithms <ul style="list-style-type: none"> - Clustering - Association Rules - Principal Component Analysis



คอร์สเราโฟกัสที่ supervised learning

Supervised Learning	Unsupervised Learning
มี Features (x) และ Labels (y)	มี Features (x) แต่ไม่มี Labels (y)
เป้าหมายคือการทำนาย	เป้าหมายคือการสรุปผล
ตัวอย่าง Algorithm: Regression, Classification	ตัวอย่าง Algorithm: Clustering, Association Rules (Market Basket Analysis/Apriori), Principal Component Analysis

-One Hot Encoding (แปลง Parameter จากไม่ใช่ตัวเลขเป็นตัวเลข ใช้กับ Algorithm ประเภท Decision Tree)

1. Thailand = 1
2. UK = 2
3. USA = 3

Thailand	UK	USA
1	0	0
0	1	0
0	0	1

*การ Train Model ใน ML เราจะต้องแปลงข้อมูลให้เป็นตัวเลข

One Hot Encoding in R:

```
Dataset %>%  
  mutate(Thailand = ifelse(country == "TH", 1, 0))
```

<https://datatricks.co.uk/one-hot-encoding-in-r-three-simple-methods>

-Dummy Variable (ใช้กับ Algorithm ประเภท Regression):

7	Dummy Variable (n-1)			
8	Country	UK	USA	Korea
9	Thailand	0	0	0
10	UK	1	0	0
11	USA	0	1	0
12	Korea	0	0	1

Problem 1:

R Problem 01

ALS อยากจะทำ market survey กับ
ลูกค้า (ทุกค่าย) ทั้งหมด 3000 คน เพื่อ
จะดูว่าตลาดคนไทยมีลูกค้าอยู่ที่ประเภท?
i.e. customer segmentation

-Unsupervised Learning

Problem 2:

R Problem 02

Gmail มีตัวกรอง email ว่าอันไหนคือ
spam อันไหนคือ ham (อีเมลดี)

-Supervised Learning

Problem 3:

R Problem 03



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

น้องอึ้ง!

-Supervised Learning

Types of Supervised Learning:

R Types of Supervised Learning

1. Regression	2. Classification
Predict numeric labels	Predict categorical labels
Examples <ul style="list-style-type: none"> - house price - customer satisfaction - personal income - how much a customer will spend 	Examples <ul style="list-style-type: none"> - 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.

Regression	Classification
ทำนาย Labels ที่เป็นตัวเลข	ทำนาย Labels ที่ไม่ใช่ตัวเลข (เป็นหมวดหมู่)
ตัวอย่าง: ราคาบ้าน, ความพึงพอใจของลูกค้า, รายได้ส่วนตัว, ลูกค้าจะยอมจ่ายเท่าไร	ตัวอย่าง: คำถามใช่หรือไม่ใช่, ทำนาย Churn, Conversion (เปลี่ยนเป็นลูกค้าเราไหม จะซื้อของๆ เราไหม), ทำนายสภาพอากาศ

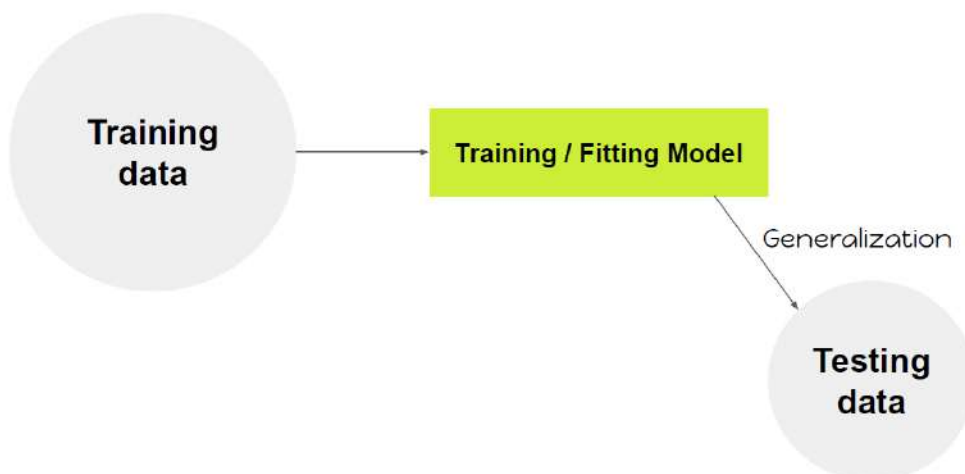
Regression	Classification
100, 200, 500, etc.	0, 1, 0, 1, 0, etc.

-Model = Algorithm(Data) เหมือนกับที่ $y = f(x)$

-ถ้า Data คือ Input, Model คือ Output, Algorithm ก็จะเปรียบเสมือน Function ที่แปลง x ให้เป็น y

-เราสามารถ Split Data ใน Excel ได้ด้วยการสร้าง Column ชื่อ Randomize แล้วกำหนดค่าใน Cell ของ column นั้นเป็น =RAND() ให้หมด จากนั้น ให้ Filter column ชื่อ Randomize ที่เราสร้างเพื่อแบ่งข้อมูล (Sort จากค่า Randomize มากสุดไปน้อยสุด แล้วแบ่งข้อมูลออกเป็นสองส่วน เช่น มีข้อมูล 32 rows แบ่ง 50:50 ก็คือ 16 rows แรกมาไว้เป็น Train Dataset แล้ว 16 rows หลังก็จะเป็น Test Dataset ของเรา เป็นต้น)

Generalization

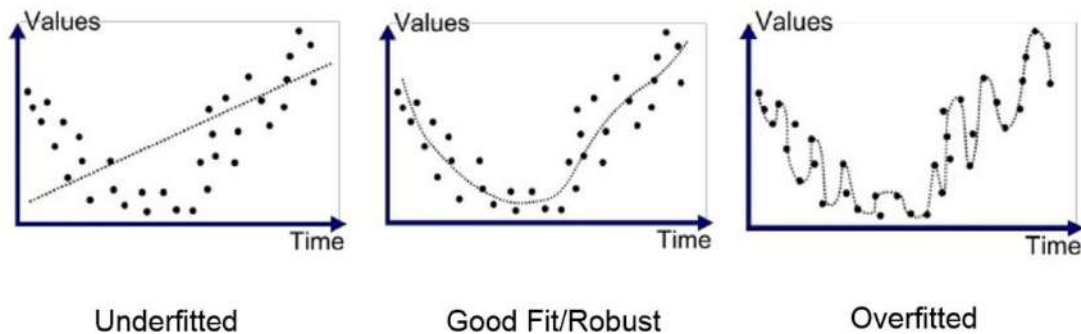


เราต้องถามคำถามนี้เสมอ
โมเดลที่เราสร้างขึ้นมาเอาไปใช้จริงได้หรือไม่? i.e.
ความถูกต้องของโมเดลกับ test data เป็นเท่าไร

Overfitting: Model ที่เราสร้างขึ้นมา Fit กับข้อมูล Train มากเกินไปจนไม่สามารถนำไปใช้กับ Test / Unseen data ได้ (หลอสนามจริง สิงห์สนามซ้อม)

Golden Rule: เราจะไม่ทดสอบ Model ด้วยข้อมูลชุดเดิมที่ใช้ Train Model เช่น เราจะไม่ใช้ Training Data วัตถุประสงค์ว่า Model ของเราทำงานดีไหม? Data ที่ใช้วัดผลต้องเป็น Unseen Data เป็นข้อมูลที่ Model ไม่เคยเห็นมาก่อน

R Our goal is in the middle -> Just Right



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

Underfit/Overfit = Not Good!

Good Fit/Robust = The Best!

Resampling = Golden Standard

1. Leave One Out CV (ทำซ้ำไปเรื่อย ๆ จนกว่าจะ Train Model ครบ n รอบ (ตามจำนวน n) แล้วหาค่าเฉลี่ย Error หรือ Accuracy ของ Model ทั้งหมด) *นานมาก
2. Bootstrap (Sampling with replacement ใช้การสุ่มซ้ำ n = 1000 เหมือน Full Dataset)
3. K-Fold Cross Validation (Train Model เสร็จไวสุด ได้ผลใกล้เคียงกับ Bootstrap) *Golden Standard

*ปกติ K-Fold Cross Validation นิยมใช้ K = 5 หรือ K = 10

R 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**

-LOOCV Train Model มาสุด รองลงมาเป็น Bootstrap และ K-Fold Cross Validation ตามลำดับ

K-Fold Cross Validation in R

```
library(tidyverse)
library(caret)

#Train Test Split
#1. Split Data
#2. Train Data
#3. Score
#4. Evaluate

glimpse(mtcars)

#Split Data 80%: 20%
train_test_split <- function(data, train_ratio = 0.7) {
  set.seed(42)
  n <- nrow(data)
  id <- sample(1 : n, size = train_ratio * n)
  train_data <- data[id, ]
  test_data <- data[-id, ]
  return( list(train = train_data, test = test_data) )
}

set.seed(42)
split_data <- train_test_split(mtcars, 0.8)
train_data <- split_data$train
test_data <- split_data$test ##split_data[["test"]]

#Train Model
model <- lm(mpg ~ hp + wt + am, data = train_data)

#Score Model
mpg_pred <- predict(model, newdata = test_data)

#Evaluate Model with Metrics
#MAE, MSE, RMSE

#Mean Absolute Error
MAE <- function(actual, prediction) {
  abs_error <- abs(actual - prediction)
  mean(abs_error)
}

#Mean Squared Error
MSE <- function(actual, prediction) {
  sq_error <- (actual - prediction) ** 2
  mean(sq_error)
}

#Root Mean Squared Error
```

```

RMSE <- function(actual, prediction) {
  sq_error <- (actual - prediction) ** 2
  sqrt(mean(sq_error))
}

#Caret = Classification And REgression Tree

#Train Test Split
#1. Split Data
split_data <- train_test_split(mtcars, 0.7)
train_data <- split_data[[1]]
test_data <- split_data[[2]]

#2. Train Data
#mpg = f(hp, wt, am)
#method = Algorithm
set.seed(42)

#Change Resampling Method (LOOCV, boot, cv) *USE cv
ctrl <- trainControl(
  method = "cv", #Golden Standard
  number = 5, #K = 5
  verboseIter = TRUE
)

#Linear Regression Model
lm_model <- train(mpg ~ hp + wt + am, data = train_data, method = "lm", trControl = ctrl)

#Random Forest Model
rf_model <- train(mpg ~ hp + wt + am, data = train_data, method = "rf", trControl = ctrl)

#K-Nearest Neighbors Model
knn_model <- train(mpg ~ hp + wt + am, data = train_data, method = "knn", trControl = ctrl)

#In this scenario: rf > lm > knn

#3. Score
p <- predict(model, newdata = test_data)

#4. Evaluate
RMSE(test_data$mpg, p)

#5. Save Model
saveRDS(model, "linear_regression_v1.RDS")

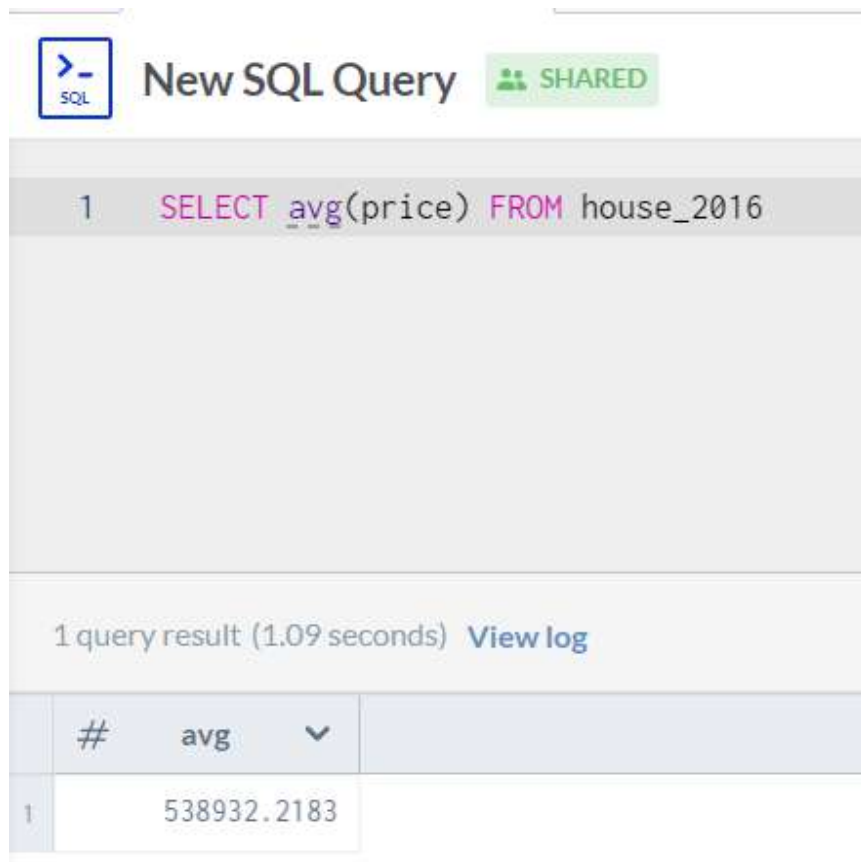
#On friend's PC
#Read model into R environment
(new_cars <- data.frame(
  hp = c(100, 150, 250),
  wt = c(1.25, 2.2, 2.25),
  am = c(0, 1, 1)
))

model <- readRDS("linear_regression_v1.RDS")

new_cars$mpg_pred <- predict(model, newdata = new_cars)
View(new_cars)

```

-เราสามารถรัน Query ในเว็บ data.world ได้



The screenshot shows a web interface for running SQL queries. At the top, there's a header with a SQL icon, the text "New SQL Query", and a "SHARED" button. Below this is a text area containing the SQL query: `1 SELECT avg(price) FROM house_2016`. Under the query, it says "1 query result (1.09 seconds)" and a "View log" link. At the bottom, there's a table with the results of the query.

#	avg
1	538932.2183

Homework Part 1

การบ้าน part 1 ให้ลองสร้าง regression model ทำนายราคาบ้านที่อินเดีย โดยใช้ข้อมูล dataset จาก data.world นะครับ ลิ้งด้านล่าง

<https://data.world/dataindianset2000/house-price-india>

สมัคร free account แล้วดาวน์โหลด Excel/ CSV ออกมาได้เลย import เข้าโปรแกรม R แล้วเขียนโมเดลด้วย `caret` เลือกตัวแปร 3-5 ตัว

Tip - ก่อนเราจะเทรนโมเดล ลอง visualize คอลัมน์ `price` จะเห็นว่าข้อมูลมันเบ้ขวามากๆ ถ้าเราอยากจะทำให้นมัน normal ขึ้น สามารถใช้ฟังก์ชัน `log(price)` เพื่อปรับ distribution ให้มีการกระจายตัวดีขึ้นได้ (เทคนิคนี้ในงาน ML เราใช้กันบ่อยๆ log transformation)

✅ สำหรับการทำ log คอลัมน์นั้นต้องห้ามมีเลขศูนย์ หรือค่าติดลบนะครับ ตัวอย่างเช่นถ้า `price` มีค่าศูนย์อยู่ด้วยให้เราใช้ฟังก์ชัน `log(price+1)` แทน เหมือนเรา +1 เข้าไปทั้งคอลัมน์

<https://data.world/dataindianset2000/house-price-india>

Dataset:

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/4bb373e9-c20b-4db4-b664-925091744ba8/House_Price_India.xlsx

My Homework Code:

```
#Activate library
library(tidyverse)
library(caret)
library(readxl)

#Import data set
house_price_data <- read_xlsx("House Price India.xlsx")
glimpse(house_price_data)
View(house_price_data)

#Basic data visualization
ggplot(house_price_data, aes(Price)) +
  geom_histogram()

#The Price column is skewed positively
#Fix skewness by using log transformation
house_price_data_log <- house_price_data %>%
  mutate(Price_log = log(Price + 1))

View(house_price_data_log)

#Data viz after fixing the skewness
ggplot(house_price_data_log, aes(Price_log)) +
  geom_histogram()

#Price column distribution is now better

#Rename columns
house_price_renamed <- house_price_data_log %>%
  rename("Distance_from_the_airport" = "Distance from the airport" ,
        "Area_of_the_house_excluding_basement" = "Area of the house(excluding basement)",
        "Area_of_the_basement" = "Area of the basement",
        "Grade_of_the_house" = "grade of the house",
        "Condition_of_the_house" = "condition of the house")

##ML Time!
#1. Split data

#train_test_split function
train_test_split <- function(data, train_ratio = 0.7) {
  set.seed(42)
```



```

n <- nrow(data)
id <- sample(1 : n, size = train_ratio * n)
train_data <- data[id, ]
test_data <- data[-id, ]
return( list(train = train_data, test = test_data) )
}

#Split data into train data and test data
set.seed(42)
split_data <- train_test_split(house_price_renamed, 0.8)
train_data <- split_data$train
test_data <- split_data$test

#2. Train data
#Set train control parameter
ctrl <- trainControl(
  method = "cv",
  number = 5,
  verboseIter = TRUE
)

#Training 3 models
#2.1 Linear Regression
lm_model <- train(Price_log ~ Distance_from_the_airport +
  Area_of_the_house_excluding_basement +
  Area_of_the_basement +
  Grade_of_the_house +
  Condition_of_the_house,
  data = train_data,
  method = "lm",
  trControl = ctrl)

#2.2 Random Forest
rf_model <- train(Price_log ~ Distance_from_the_airport +
  Area_of_the_house_excluding_basement +
  Area_of_the_basement +
  Grade_of_the_house +
  Condition_of_the_house,
  data = train_data,
  method = "rf",
  trControl = ctrl)

#2.3 K-Nearest Neighbors
knn_model <- train(Price_log ~ Distance_from_the_airport +
  Area_of_the_house_excluding_basement +
  Area_of_the_basement +
  Grade_of_the_house +
  Condition_of_the_house,
  data = train_data,
  method = "knn",
  trControl = ctrl)

#3. Score
p_lm <- predict(lm_model, newdata = test_data)
p_rf <- predict(rf_model, newdata = test_data)
p_knn <- predict(knn_model, newdata = test_data)

#4. Evaluate
#Evaluate Model with Metrics

```

```

#MAE, MSE, RMSE

#Mean Absolute Error
MAE <- function(actual, prediction) {
  abs_error <- abs(actual - prediction)
  mean(abs_error)
}

#Mean Squared Error
MSE <- function(actual, prediction) {
  sq_error <- (actual - prediction) ** 2
  mean(sq_error)
}

#Root Mean Squared Error
RMSE <- function(actual, prediction) {
  sq_error <- (actual - prediction) ** 2
  sqrt(mean(sq_error))
}

##train - test comparison
#lm
lm_model
MAE(test_data$Price_log, p_lm)
MSE(test_data$Price_log, p_lm)
RMSE(test_data$Price_log, p_lm)

#rf test
rf_model
MAE(test_data$Price_log, p_rf)
MSE(test_data$Price_log, p_rf)
RMSE(test_data$Price_log, p_rf)

#knn test
knn_model
MAE(test_data$Price_log, p_knn)
MSE(test_data$Price_log, p_knn)
RMSE(test_data$Price_log, p_knn)

#5. Save model
saveRDS(lm_model, "lm_model_v1.RDS")
saveRDS(rf_model, "rf_model_v1.RDS")
saveRDS(knn_model, "knn_model_v1.RDS")

```

Part 2

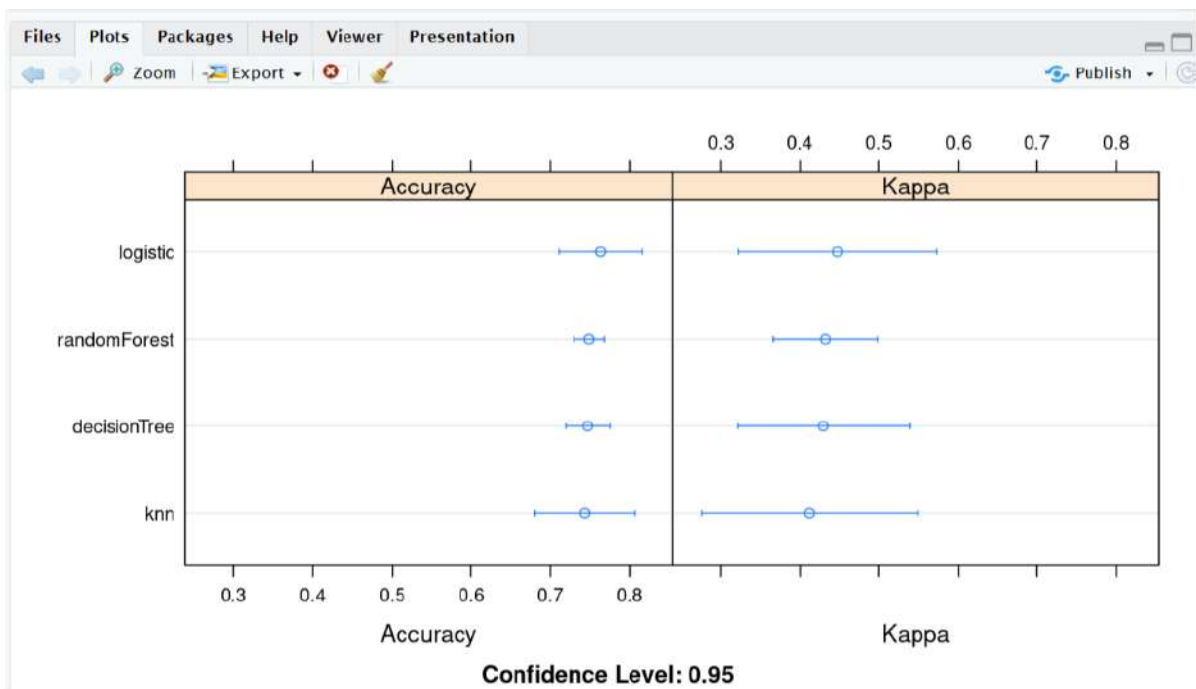
Worksheet Part 2:

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/422fb661-f81f-4b4c-95a2-897b0743e24b/worksheet_ml2.xlsx

Tip - ตอนแอตสร้าง `resamples` เปรียบเทียบหลายๆโมเดล ถ้าอยากจะ plot chart ให้ใช้ฟังก์ชัน `dotplot()` ได้เลยนะครับ จะได้ output ตามรูปด้านล่าง ไว้เปรียบเทียบโมเดล

```
Console Terminal Background Jobs
R 4.2.2 · /cloud/project/
logistic      0.3378033 0.3718834 0.4268456 0.4474727 0.5198786
decisionTree  0.3030597 0.3729337 0.4846121 0.4297888 0.4879884
randomForest  0.3399621 0.4366967 0.4518717 0.4326034 0.4614479
      Max. NA's
knn      0.5320048 0
logistic  0.5809524 0
decisionTree 0.5003502 0
randomForest 0.4730389 0

> dotplot(result)
> |
```



No Free Lunch: ไม่มีโมเดลไหนเก่งที่สุดและสามารถตอบโจทย์ได้ทุกปัญหา ความเก่งของโมเดล ขึ้นกับข้อมูล

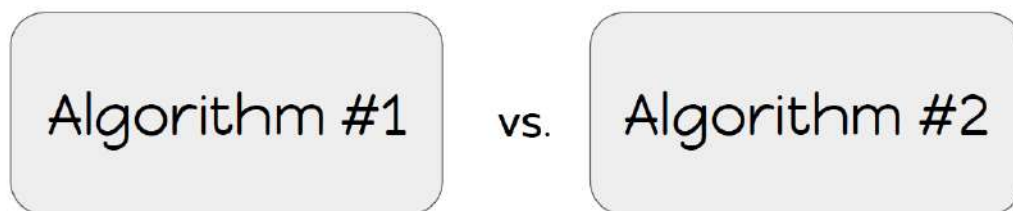
No Free Lunch แปลว่า “ไม่มีโมเดลไหนเก่งที่สุด และสามารถตอบโจทย์ได้ทุกปัญหา”

ถ้ามีใครถามว่าโมเดลไหนเก่งที่สุด?

ให้ตอบว่า “It depends” (ขึ้นอยู่กับข้อมูล)

ความท้าทายของ ML คือการหาโมเดลที่ดีที่สุดสำหรับปัญหาที่เราทำล้งแก้

Occam's Razor: Choose a simpler model if performances are the same (เลือกตัวที่ Train ง่ายสุด Implement ง่ายสุด และนำไปใช้งานจริงได้ง่ายที่สุด)



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

How to choose a model?:

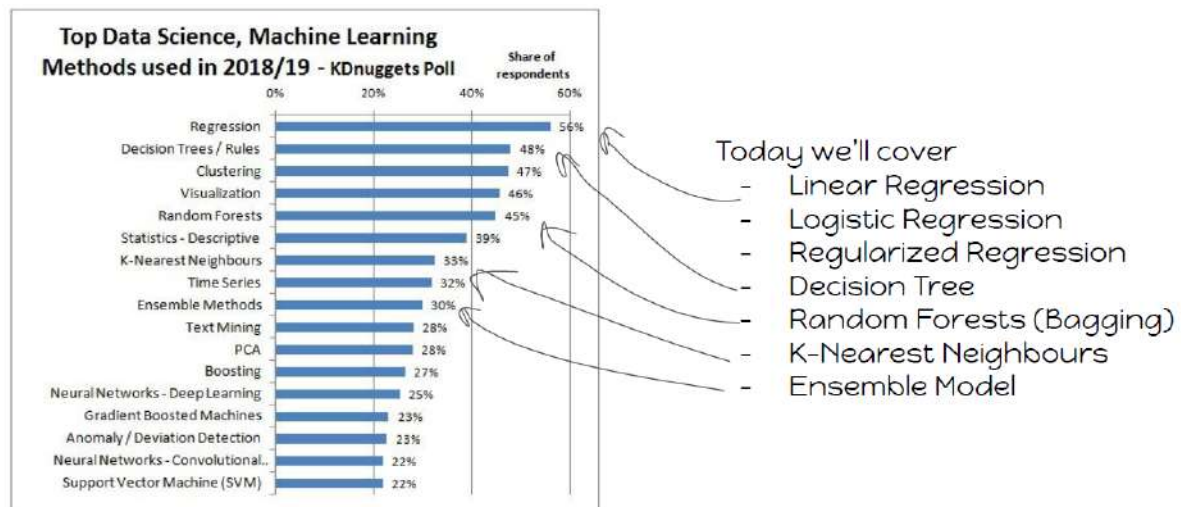
1. Regression or Classification?
2. High accuracy or High interpretability?

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.

*Trade-off between accuracy and explainability

ในเชิงธุรกิจ (Business) การใช้ Model ประเภท Regression จะสามารถอธิบายผลได้ง่ายกว่า Model ที่เป็น Black Box (อธิบายการทำงานได้ยากกว่า)



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

Marketing Mix Modeling

Marketing mix modeling (MMM) is statistical analysis such as **multivariate regressions** on **sales** and **marketing time series** data to estimate the impact of various marketing tactics (**marketing mix**) on sales and then forecast the impact of future sets of tactics. It is often used to optimize **advertising mix** and promotional tactics with respect to sales revenue or profit.

The techniques were developed by **econometricians** and were first applied to **consumer packaged goods**, since manufacturers of those goods had access to accurate data on sales and marketing support.

[citation needed] Improved availability of data, massively greater computing power, and the pressure to measure and optimize marketing spend has driven the explosion in popularity as a marketing tool. [citation needed] In recent times MMM has found acceptance as a trustworthy marketing tool among the major consumer marketing companies.

KNN (K-Nearest Neighbors)

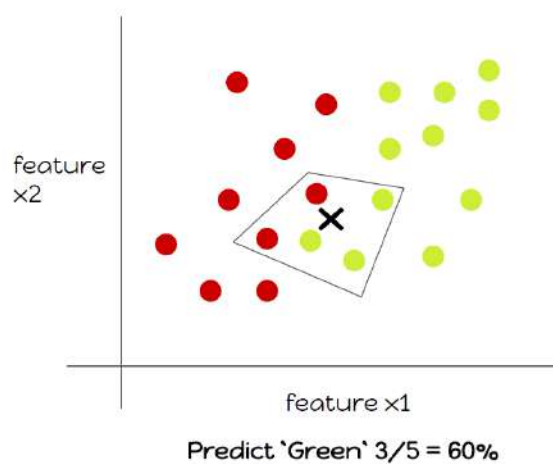
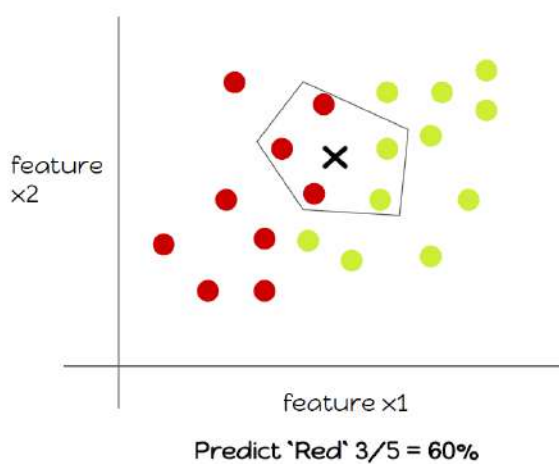
-หลักการคือ วัดระยะห่างระหว่างข้อมูลทุกจุดกับข้อมูลที่เราต้องการทำนาย แล้วดูว่าข้อมูลที่เราต้องการทำนายอยู่ใกล้ Label ไหนมากกว่ากัน

R Euclidean Distance

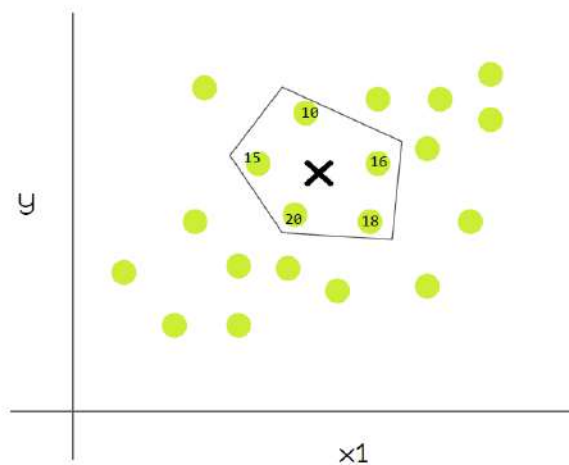
$$d = \sqrt{(x1 - x2)^2 + (y1 - y2)^2}$$

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

R We use majority vote to assign label



R We use average value for regression problem



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

1. Choose K
2. Compute Distance
3. Majority vote for classification OR Average for regression

*ต้อง Standardize ข้อมูลก่อนการ Train Model แบบ KNN เสมอ (ใส่ `preProcess = c("center", "scale")`)

-การ Tuning parameter ต่าง ๆ ในการ Train Model ถือเป็นสิ่งสำคัญมาก เพราะการปรับ Knob ไปยังตำแหน่งที่ถูกต้องจะทำให้ Model เราให้ผลดีขึ้น

-Linear Regression ก็ควรทำ Center and Scale (Normalization) เช่นเดียวกัน *ทุก Model ควรจะ Center and Scale มีแต่ได้ไม่มีเสีย

Random Search

R 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



-Set Tune Length ให้คอมพิวเตอร์เลือกค่า K ให้เราเอง ต่างจาก Tune Grid ที่เราเลือกค่า K ให้คอมพิวเตอร์

Key Learning:

1. KNN เข้าใจง่าย ทำงานได้ดีถ้า Feature ไม่เยอะมาก
2. KNN ใช้ได้ทั้ง Regression และ Classification
3. K ใน KNN ค่าค่า Hyperparameter ที่เราสามารถเปลี่ยนได้ (Knob)
4. เราจะเลือก K ที่ทำให้ Train Model แล้วได้ค่า RMSE ที่ต่ำที่สุด
5. แต่การที่เราได้ค่า RMSE ที่ต่ำที่สุดมา ไม่ได้หมายความว่าโมเดลเราจะทำนาย Test Data ได้ดี ต้องเอาไปทดสอบอีกที

Classification VS Regression

Classification

```
set.seed(42)

ctrl <- trainControl(method = "cv",
  number = 5)

model <- train(
  y ~ .,
  data = df,
  method = "knn",
  metric = "Accuracy",
  trControl = ctrl
)
```

Regression

```
set.seed(42)

ctrl <- trainControl(method = "cv",
  number = 5)

model <- train(
  y ~ .,
  data = df,
  method = "knn",
  metric = "RMSE",
  trControl = ctrl
)
```

Same interface, different metrics

Classification Interfaces

Classification - ROC Sens Specs

```
set.seed(42)

ctrl <- trainControl(
  method = "cv",
  number = 5,
  summaryFunction = twoClassSummary,
  classProbs = TRUE)

model <- train(
  y ~ .,
  data = df,
  method = "knn",
  metric = "ROC",
  trControl = ctrl
)
```

Classification - AUC Precision Recall F1

```
set.seed(42)

ctrl <- trainControl(
  method = "cv",
  number = 5,
  summaryFunction = prSummary,
  classProbs = TRUE)

model <- train(
  y ~ .,
  data = df,
  method = "knn",
  metric = "AUC",
  trControl = ctrl
)
```

-ไม่ใช่ทุกปัญหาที่เหมาะสมกับการใช้ Accuracy โดยเฉพาะ Imbalanced Classification

What is Kappa? (Normalized Accuracy)

 CHAYAN KATHURIA · POSTED 2 YEARS AGO IN [GENERAL](#)

What's Kappa Score? 🤔

Kappa Score is an interesting, but an underused metric. Also known as Cohen's Kappa Statistic on the name of Jacob Cohen, Kappa Score is very useful when evaluating classification models. How? Read on. 📖

The fundamental concept behind the Kappa Score is it measures the amount of "agreement" between two values. In classification, one of this is the predicted value and other is the ground truth. 🗨️

Kappa score takes into consideration not 1, but 2 different accuracy measures. One is the usual Predictions accuracy, and the other one is the Expected accuracy. 🤖

The expected accuracy is the accuracy which can be attained by any random predictions.

Kappa Score is calculated as:

$$K = (\text{Predicted accuracy} - \text{Expected accuracy}) / (1 - \text{Expected accuracy})$$

So, if $K = 0.4$, and expected accuracy is 50%, you can say that your classifier is performing 40% better than the random predictions, meaning a prediction accuracy of 70%. 🗨️

However, if your expected accuracy itself was 70%, and the model also gave 70% accuracy, K will be 0. 📉

A low value of K means, a low level of "agreement" between the classifier and the ground truth. 🗨️

Kappa Score can also be used to compare the performance of 2 models in the same fashion. 🗨️

Link: <https://www.kaggle.com/general/185898>

AUC = พื้นที่ใต้กราฟของ Precision-Recall Curve (ยิ่งเยอะยิ่งดี) ใช้วัดว่าทำนายได้ดีทั้ง Precision และ Recall หรือไม่

ก่อนเราจะทำการ Subset Data เพื่อนำไป Train Test Split เราจะต้อง Relevel ในส่วนของ Factor ข้อมูลที่เราต้องการจะทำนาย เพื่อให้ผลของ Train Model และ Confusion Matrix มีความสอดคล้องกัน (ทำให้ Focus ไปในทิศทางที่ถูกต้อง)







```
data("PimaIndiansDiabetes")
df <- PimaIndiansDiabetes

#Contrasts positive and negative
df$diabetes <- fct_relevel(df$diabetes, "pos")
```

Sigmoid Function สามารถบีบตัวเลขให้อยู่ระหว่าง 0 กับ 1 ได้
-method = "glm"

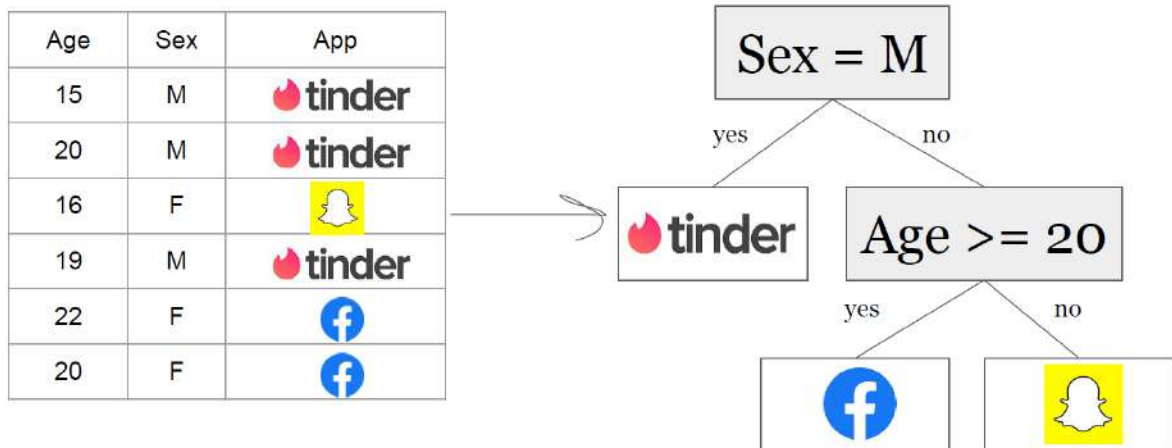
Decision Tree (Rule-based Algorithm)

How decision tree work?

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

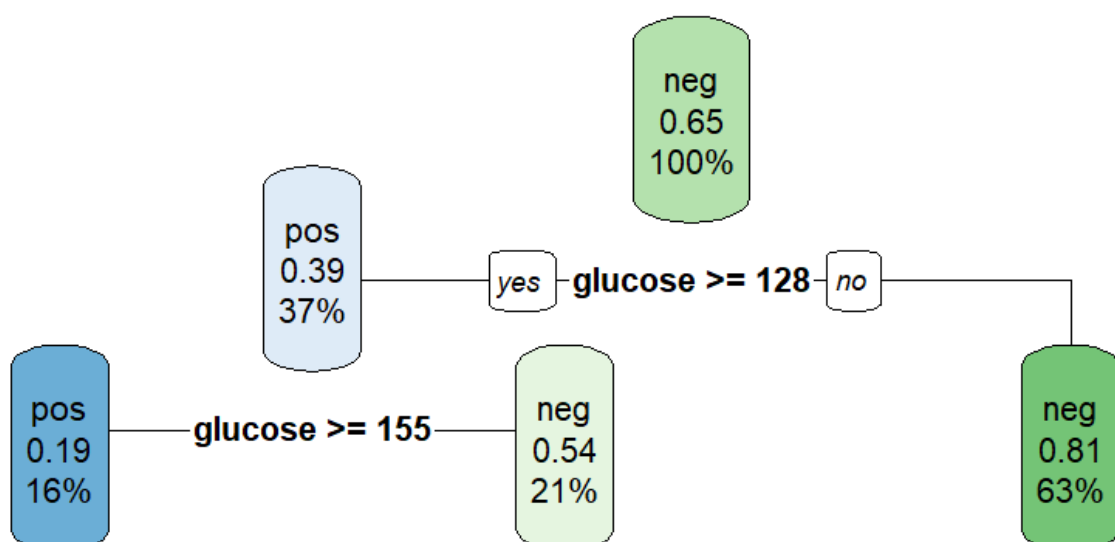
เวลาเราสร้าง decision tree เราถามคำถาม yes/no question ทีละข้อ

i.e. feature ใช้แบ่ง App ได้ดีที่สุด



*Linear Regression และ Logistic Regression จะมี Final Model ซึ่งจะบอกค่าสัมประสิทธิ์ (Coefficient) ของแต่ละ Parameter ให้เรารู้ แต่ Decision Tree จะบอกเป็น If-Else แทน Nested เป็น Layer

```
logit_model$finalModel
rpart.plot(tree_model$finalModel)
```



ข้อดีของตระกูล Tree-based คือเราไม่จำเป็นต้อง Normalize ข้อมูลก่อนนำมาสร้างโมเดล

Random Forest

-หลักการคือ สร้าง Decision Tree เป็นร้อย ๆ ต้นแบบ Random (Uncorrelated Decision Tree) แล้วจับรวมกันเพื่อทำนายผลลัพธ์ (หลักการเดียวกับ KNN)



We grow hundreds of **uncorrelated (decision) trees**

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

R Teamwork (Bagging)

Bootstrap + mtry hyperparameter

All features

X1	X2	X3	X4	X5
----	----	----	----	----

X1	X3	X4
----	----	----

Tree 1

X2	X3	X5
----	----	----

Tree 2

X1	X2	X5
----	----	----

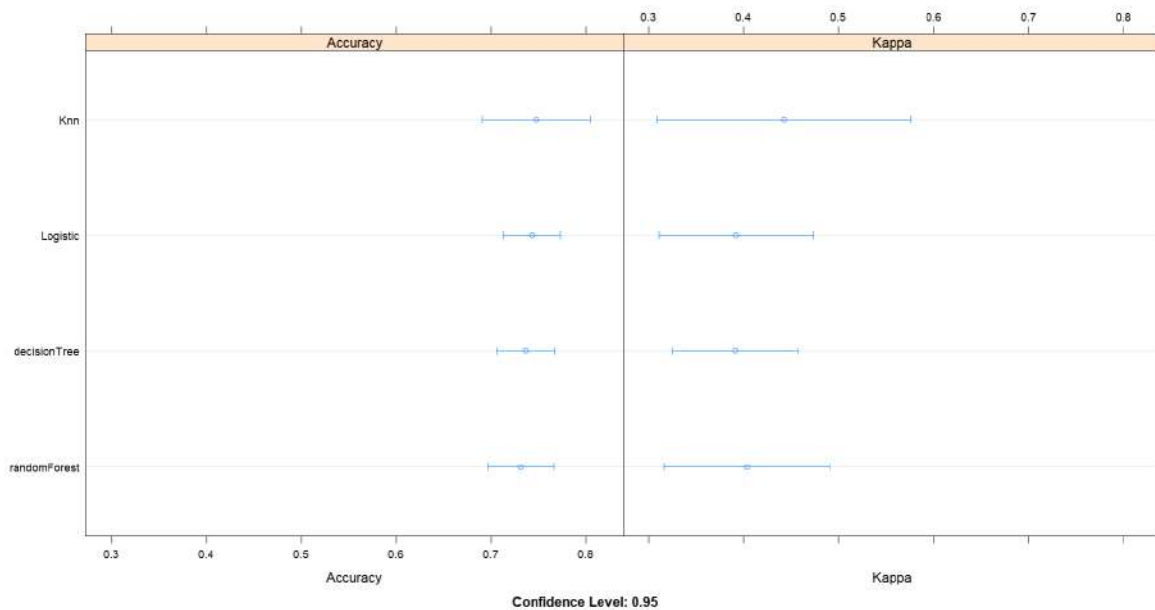
Tree 3

X2	X3	X4
----	----	----

Tree 100



Model Comparison



*ถ้าเราอยากจะ Combine Model เข้าด้วยกัน เราควรจะเลือกคู่ Model ที่มีค่า Correlation ตีลบ (ใช้ modelCor เช็ค) เรียกว่า Ensemble Model (นำ Model หลาย ๆ ตัวมารวมกันเพื่อช่วยทำนาย ผลแบบ Majority Vote)

R Ensemble Models

นำโมเดลหลายๆตัวมาช่วยกัน ทำนายผล (Majority Vote)

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

Ridge and Lasso Regression

- เราสามารถ 'ทำโทษ' ค่าสัมประสิทธิ์เพื่อลดการ Overfit ของ Model เราได้ (Lambda = Penalty)
- Lasso = Absolute ส่วน Ridge = ยกกำลังสอง

R 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

R 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

R Regularization helps reduce overfitting

$$y_hat = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + b_4x_4$$

$$y_hat = 100 + 150x_1 + 200x_2 + 120x_3 + 80x_4$$


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

-ถ้าค่า Lambda = 0 จะได้ Error เหมือน Linear Regression ปกติ

-ถ้าค่า Lambda > 0 ค่าสัมประสิทธิ์ (B) ใน Model ของเราจะลดลงเพื่อให้ fit กับ Test Data ได้ดีขึ้น

-ปกติจะตั้ง Lambda เริ่มที่ 0.1

Lambda = 0

Error is the same as Linear Regression

Lambda > 0

All coefficient (B) in the model must be **shrunk** to reduce the new error

Quick Review

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

You can save models for later usage.

Save our models for later use

```
saveRDS(model, "model.rds")

model <- readRDS("model.rds")
```

Summary

- ML = Art + Science
- No Free Lunch = Try different models
- Occam's Razor = Choose simpler model
- Use Cross Validation (CV) and Grid Search to fine-tune model
- Start with Regression or Decision Tree because they are very fast to train
- Business tends to like Regression models because they are easy to explain ideas behind them
- But, for tasks that do not require model explanation, you can use other models that are hard to explain but tend to give better results (Black Box Models)

Course 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**

ถ้าอยากรู้วิธีการใช้ caret เพิ่มเติม สามารถเข้าไปดูได้ที่ Link นี้:
<https://topepo.github.io/caret/index.html>

Code:

```
Load Library
library(tidyverse)
library(caret)
library(mlbench)
library(MLmetrics)
library(forcats)
library(rpart.plot)
library(glmnet)

#See available data
data()

#Glimpse data
data("BostonHousing")
df <- BostonHousing
glimpse(df)

#Clustering -> Segmentation
#Choose features to be considered in segmentation
subset_df <- df %>%
  select(crim, rm, age, lstat, medv) %>%
  as_tibble()

#*Normalize data first before segmentation
#Test different k-means clusters to find the best k
#k = 2 to 5
kmeans(x = subset_df, centers = 2)
kmeans(x = subset_df, centers = 3)
```

```

kmeans(x = subset_df, centers = 4)
kmeans(x = subset_df, centers = 5)

result <- kmeans(x = subset_df, centers = 3)

#Membership [1, 2, 3]
subset_df$cluster <- result$cluster

subset_df %>%
  group_by(cluster) %>%
  summarise(mean(medv))

#Equals to
result$centers

View(subset_df)

#Segment quantity depends on the Marketing Team

##KNN
df <- as_tibble(df)

#1. Split Data
set.seed(42)
n <- nrow(df)
id <- sample(1:n, size = 0.8*n)
train_data <- df[id, ]
test_data <- df[-id, ]

#2. Train Data
set.seed(42)
ctrl <- trainControl(method = "cv",
                     number = 5,
                     verboseIter = TRUE)

lm_model <- train(medv ~ crim + rm + age,
                 data = train_data,
                 method = "lm",
                 preProcess = c("center", "scale"))

#Define grid to grid search [Must be a Data Frame]
#Grid search tune hyper parameters
k_grid <- data.frame(k = c(3, 5, 7, 9, 11))

knn_model <- train(medv ~ crim + rm + age,
                  data = train_data,
                  method = "knn",
                  metric = "Rsquared",
                  tuneGrid = k_grid,
                  preProcess = c("center", "scale"),
                  trControl = ctrl)

#Tune length random search
knn_model <- train(medv ~ crim + rm + age,
                  data = train_data,
                  method = "knn",
                  metric = "Rsquared",
                  tuneLength = 2,
                  preProcess = c("center", "scale"),

```



```

        trControl = ctrl)

#3. Score Model
p <- predict(knn_model, newdata = test_data)

#4. Evaluate Model
RMSE(p, test_data$medv)

##Classification
#Diabetes data
data("PimaIndiansDiabetes")
df <- PimaIndiansDiabetes

#Contrasts positive and negative
df$diabetes <- fct_relevel(df$diabetes, "pos")

subset_df <- df %>%
  select(glucose, insulin, age, diabetes)

#1. Split Data
set.seed(42)
n <- nrow(df)
id <- sample(1:n, size = 0.8*n)
train_data <- subset_df[id, ]
test_data <- subset_df[-id, ]

#2. Train Model
set.seed(42)

ctrl <- trainControl(method = "cv",
                     number = 5,
                     verboseIter = TRUE)

(knn_model <- train(diabetes ~ .,
                  data = train_data,
                  method = "knn",
                  preProcess = c("center", "scale"),
                  metric = "Accuracy",
                  trControl = ctrl))

#3. Score Model
p <- predict(knn_model, newdata = test_data)

#4. Evaluate Model
#Manually create confusion matrix
table(test_data$diabetes, p, dnn = c("Actual",
                                     "Prediction"))

#Using function to create confusion matrix
confusionMatrix(p, test_data$diabetes,
               positive = "pos",
               mode = "prec_recall")
#Sensitivity = Recall (True Positive Rate)
#Specificity = Precision (True Negative Rate)

##Logistic Regression
set.seed(42)

ctrl <- trainControl(method = "cv",

```

```

        number = 5,
        verboseIter = TRUE)

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

##Decision Tree
set.seed(42)

ctrl <- trainControl(method = "cv",
                     number = 5,
                     verboseIter = TRUE)

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

rpart.plot(tree_model$finalModel)

##Random Forest
#Unleash the true power of Random Forest
#by NOT subsetting data
#Use bootstrap sampling
#mtry = number of features used to train model
df$diabetes <- fct_relevel(df$diabetes, "pos")

set.seed(42)
n <- nrow(df)
id <- sample(1:n, size = 0.8*n)
train_data <- df[id, ]
test_data <- df[-id, ]

set.seed(42)
mtry_grid <- data.frame(mtry = 2:8)
(rf_model <- train(diabetes ~ .,
                  data = train_data,
                  method = "rf",
                  metric = "Accuracy",
                  tuneGrid = mtry_grid,
                  trControl = ctrl))

##Compare models by creating list of all models
#Must use the same dataset (If subset = subset all)
#If using all data = use all data for all models

list_models <- list(Knn = knn_model,
                   Logistic = logit_model,
                   decisionTree = tree_model,
                   randomForest = rf_model)

result <- resamples(list_models)

summary(result)
modelCor(result)

```

```
dotplot(result)

##Ridge and Lasso
#0 = Ridge, 1 = Lasso
#Use glmnet library
glmnet_grid <- expand.grid(alpha = 0:1,
                          lambda = c(0.1, 0.2, 0.3))

(glmnet_model <- train(diabetes ~ .,
                      data = train_data,
                      method = "glmnet",
                      metric = "Accuracy",
                      tuneLength = 10,
                      trControl = ctrl))
```
