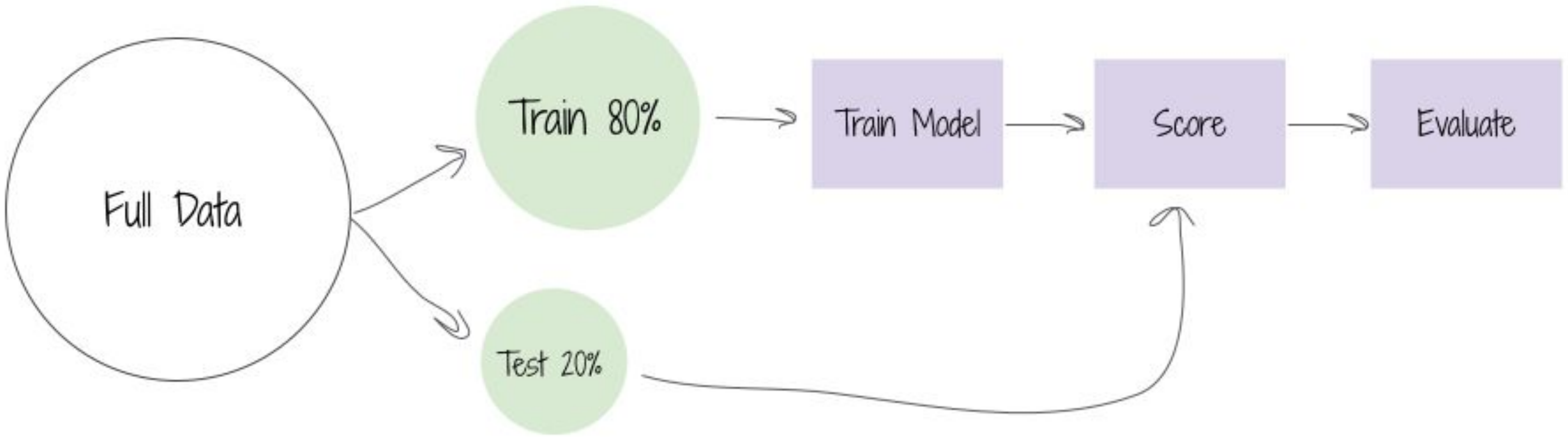
Intro to ML

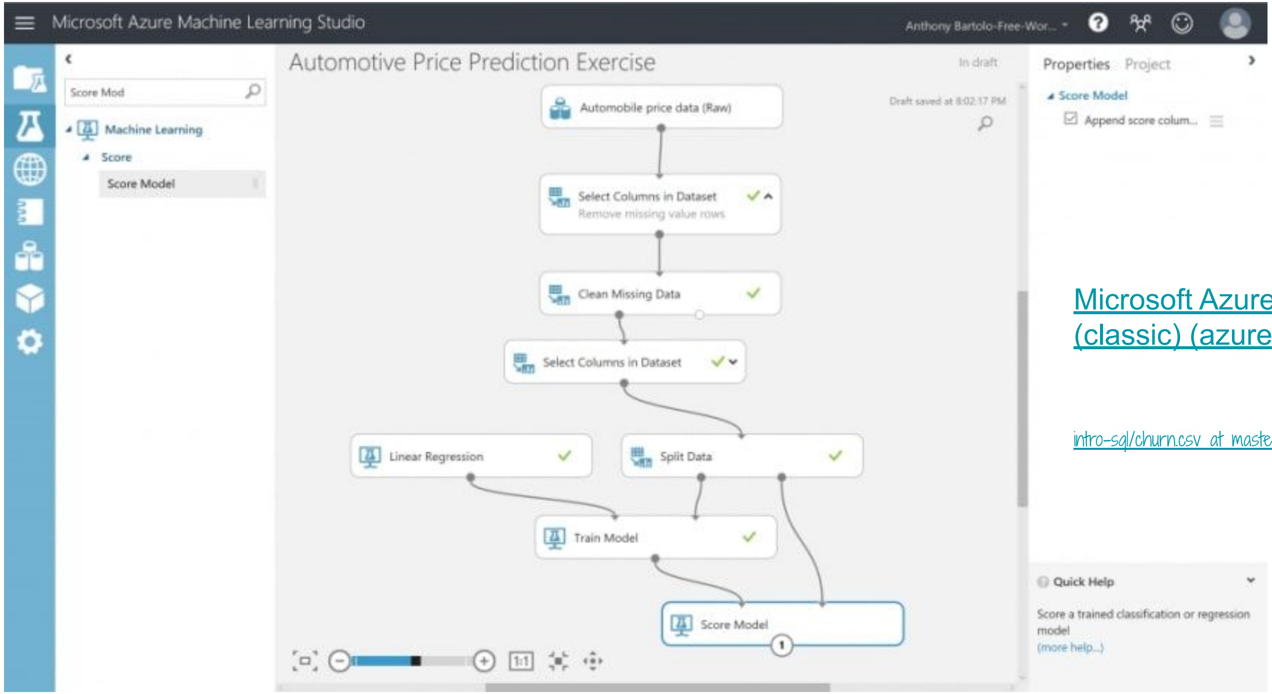
~~Live 08~~ - Data Science Bootcamp

Live 11

Simple pipeline to build ML models



Build your first model with Azure ML Studio



[Microsoft Azure Machine Learning Studio (classic) (azureml.net)]

[intro-sql/churn.csv at master · toyeiei/intro-sql (github.com)]

Example dataset

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

What is Machine Learning

Arthur Samuel (1959)

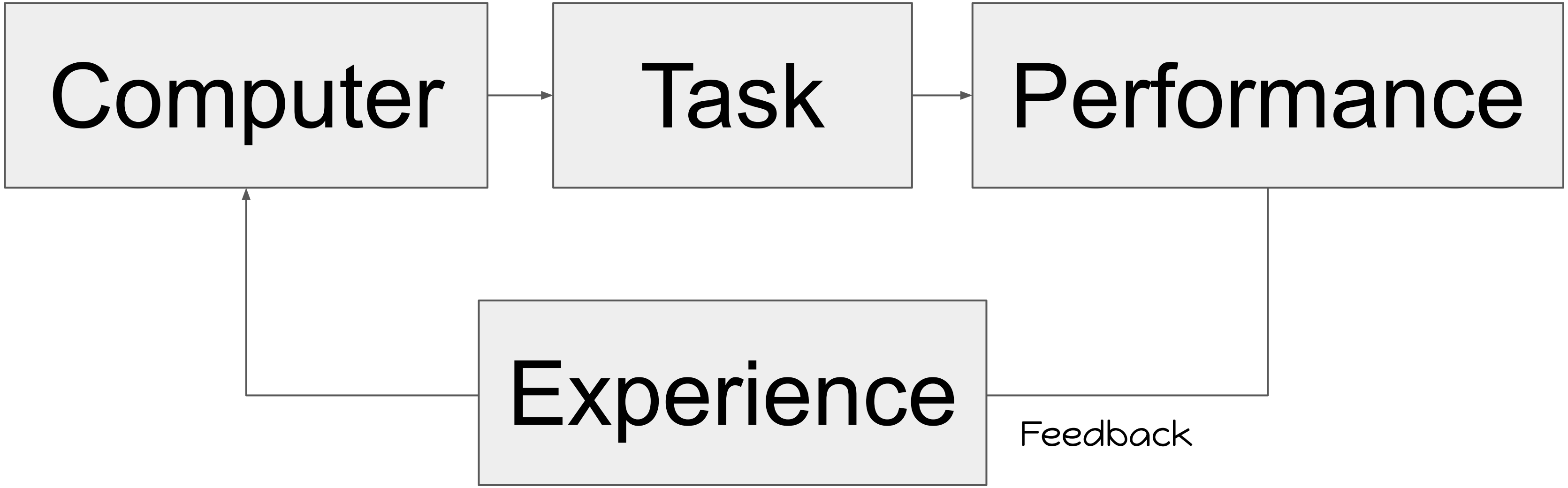
Field of study that gives computers **the ability to learn without being explicitly programmed**.

How do we learn?

Human learn from **experience**.

Computer learn from **data**.

Simple Idea



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

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	0.08829												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

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	6.998	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	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
7	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
8	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
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

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

คอร์สเราโฟกัสที่ supervised learning

Problem 01

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

Problem 02

Gmail มีตัวกรอง email ว่าอันไหนคือ spam อันไหนคือ ham (อีเมล์ดี)

Problem 03

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

น้องอิ้ง!

Types of Supervised Learning

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.

ML = Pokemon

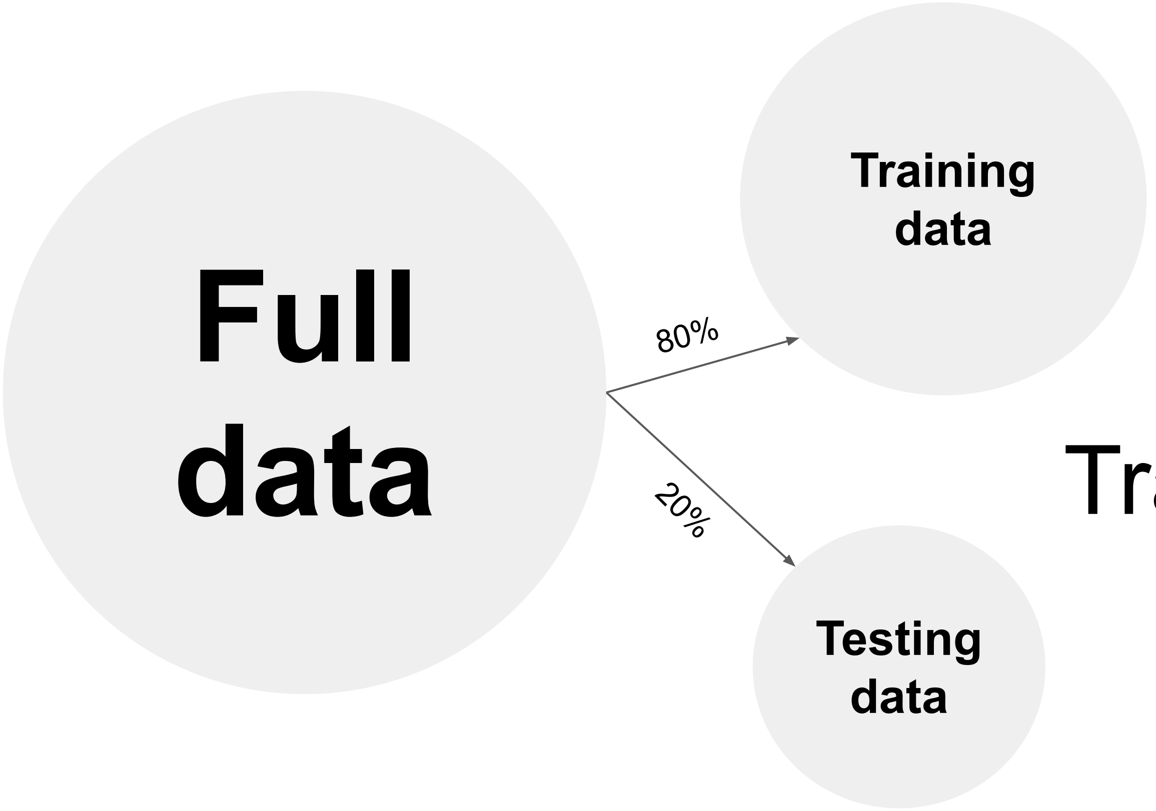
Now let’s get into the details :)

3 steps to build ML

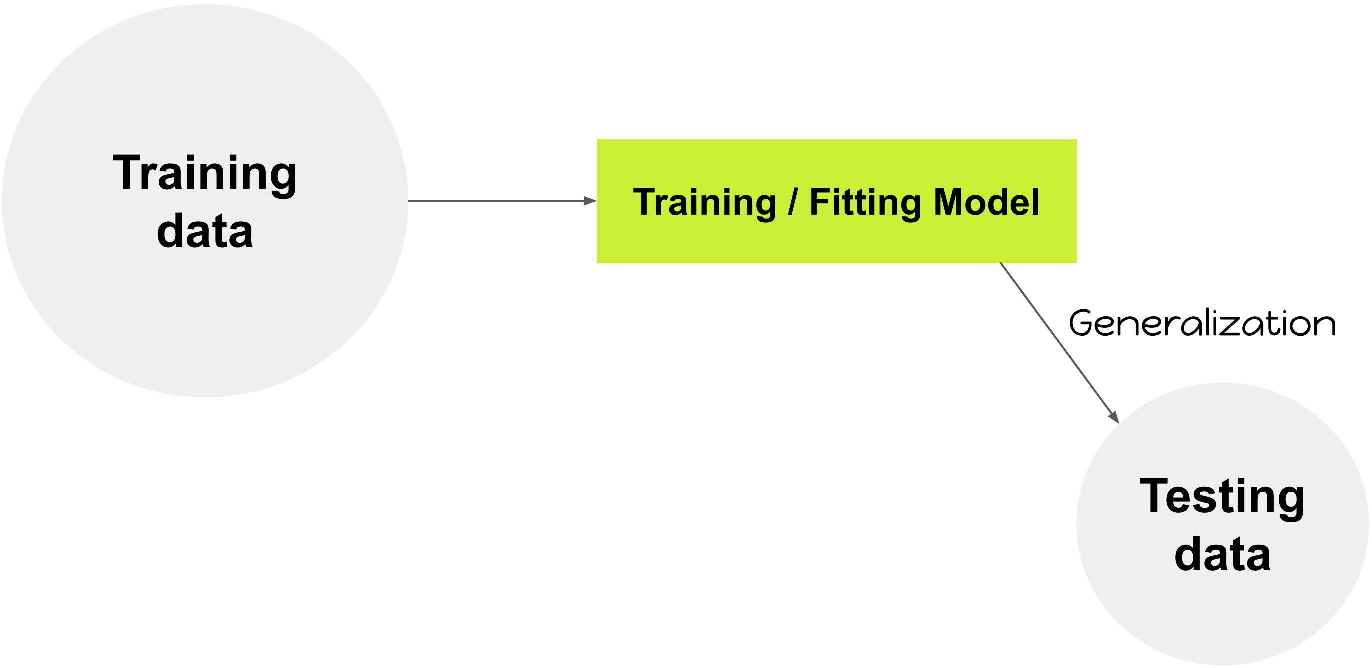
- prepare data
- train algorithm
- test/ evaluate algorithm

ML Glossary 02

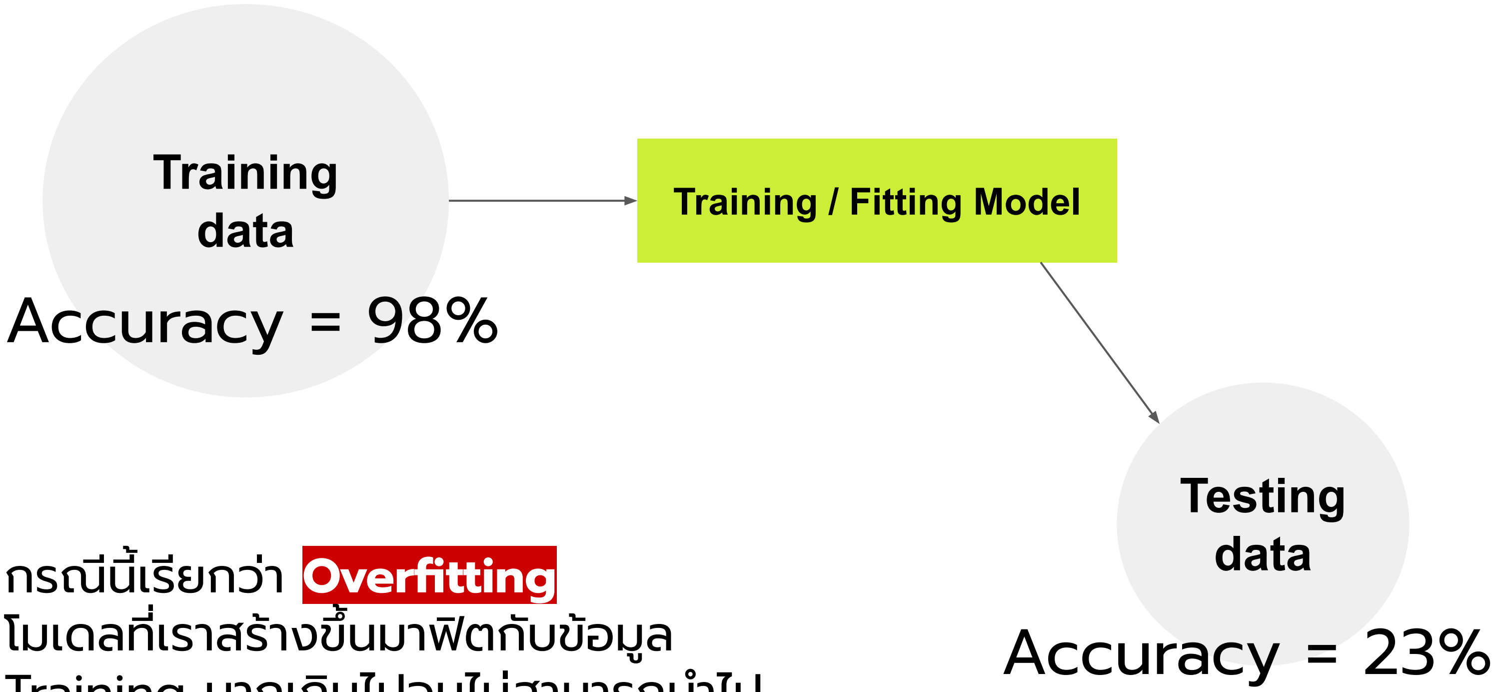
- train test split
- training set
- testing/ validation set
- overfitting



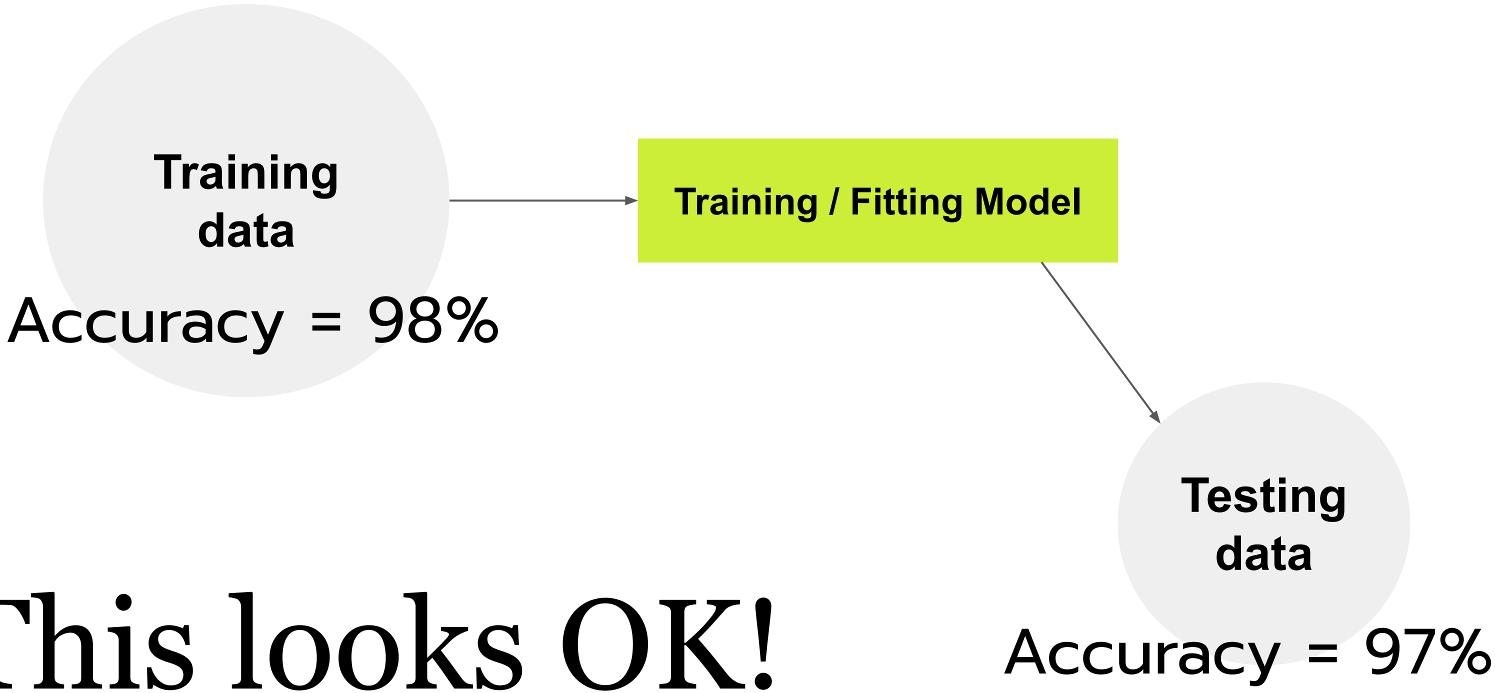
Train test split



เราต้องถามคำถามนี้เสมอ
โมเดลที่เราสร้างขึ้นมาเอาไปใช้จริงได้หรือเปล่า? i.e.
ความถูกต้องของโมเดลกับ test data เป็นเท่าไร



กรณีนี้เรียกว่า **Overfitting**
โมเดลที่เราสร้างขึ้นมาฟิตกับข้อมูล
Training มากเกินไปจนไม่สามารถนำไป
ใช้กับ Testing/ Unseen data ได้



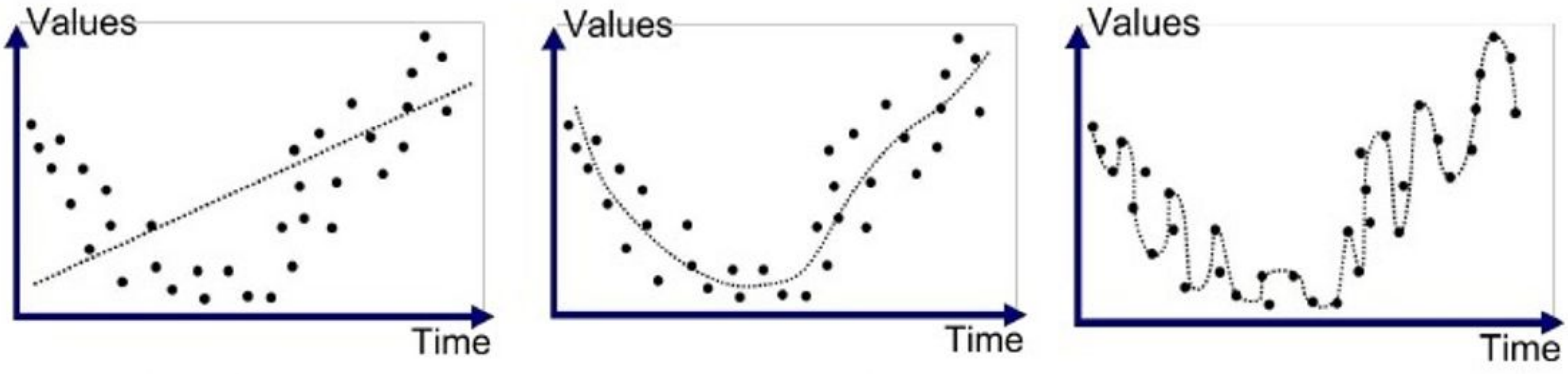
This looks OK!

Golden Rule

เราจะไม่ทดสอบโมเดลด้วยข้อมูลชุดเดิมที่ใช้เทรนโมเดล

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

Our goal is in the middle -> Just Right



Underfitted

Good Fit/Robust

Overfitted

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

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

Leave One Out CV

Full data

n=1000

Training data

n=999

Testing data

n=1

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

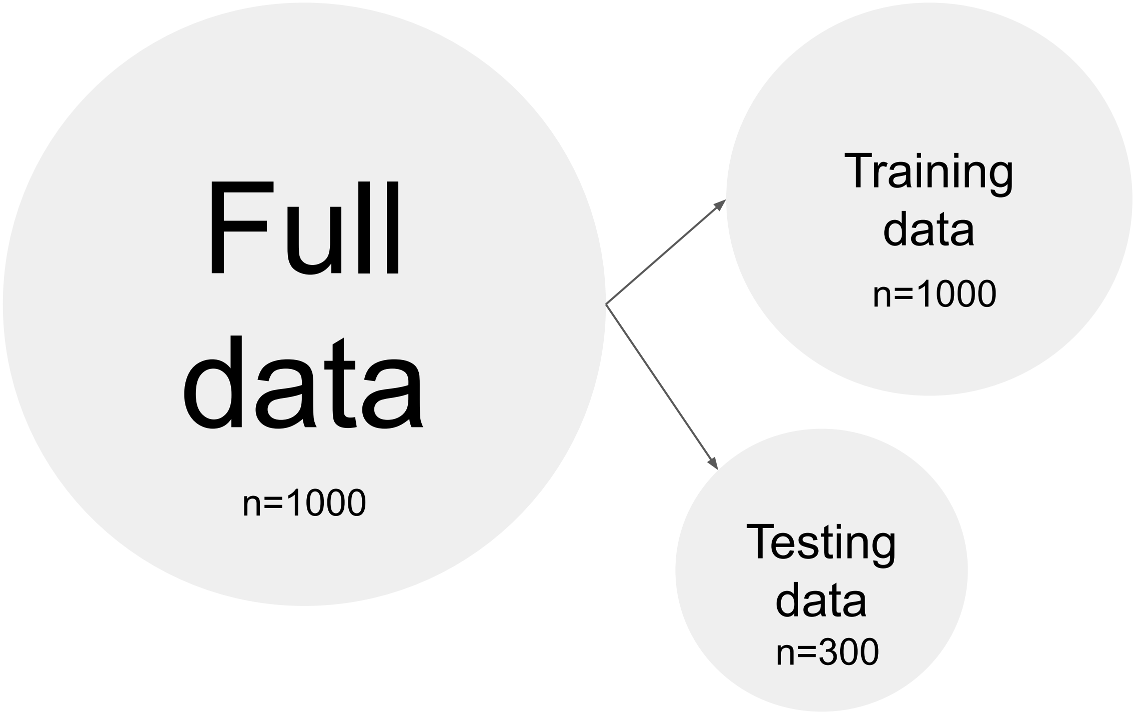
Leave One Out CV

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

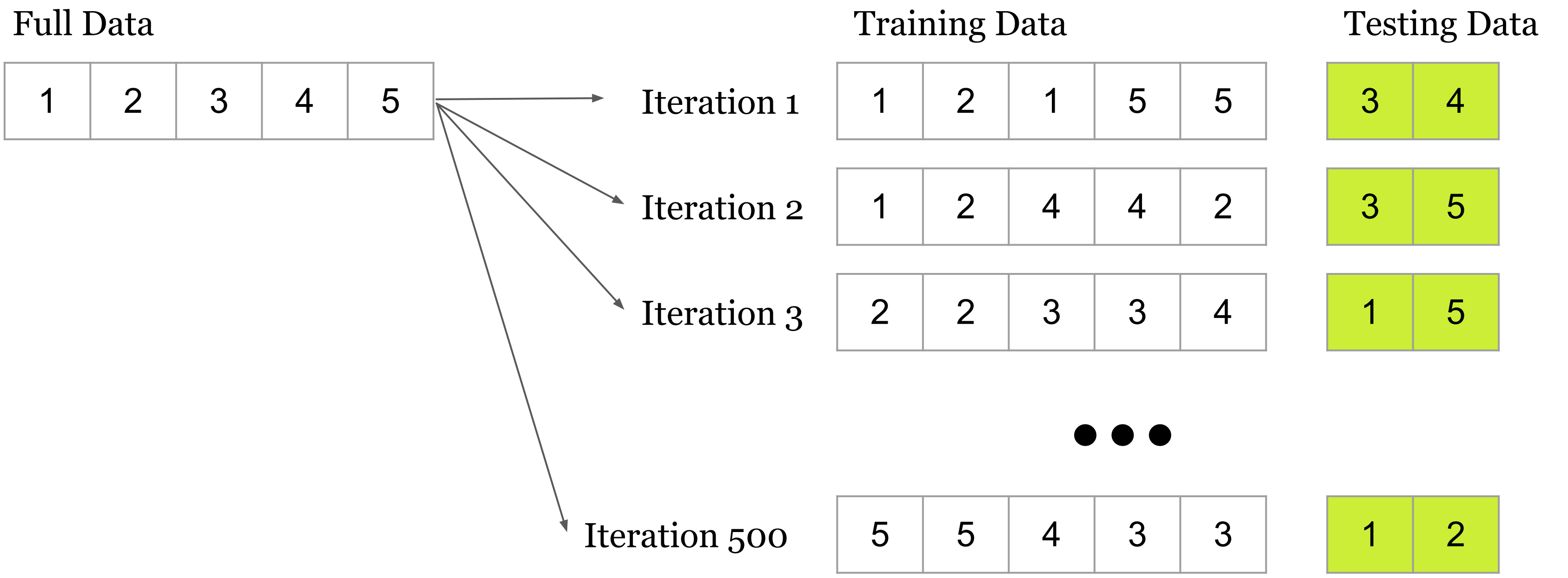
Bootstrap



Sampling with replacement

ใช้การสุ่มซ้ำ n=1000 เหมือน full dataset

Bootstrap



The error will be averaged over 500 training iterations

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**

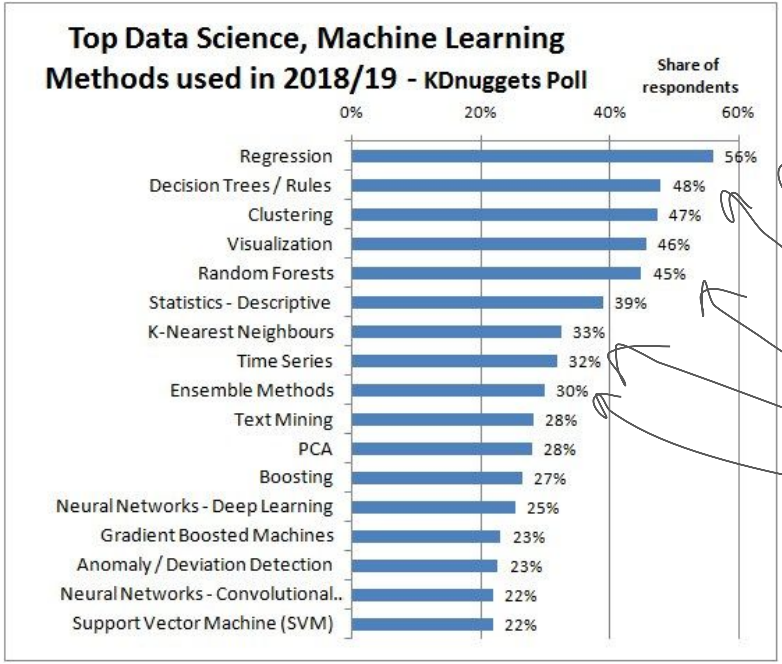
Discuss: LOOCV, Bootstrap, K-Fold ทั้งสามวิธีแตกต่างกันอย่างไร?

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

Popularity of Algorithms



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 คือการหาโมเดลที่ดีที่สุดสำหรับปัญหาที่เรากำลังแก้

Occam's Razor

Algorithm #1 vs. Algorithm #2

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

How to 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 our projects

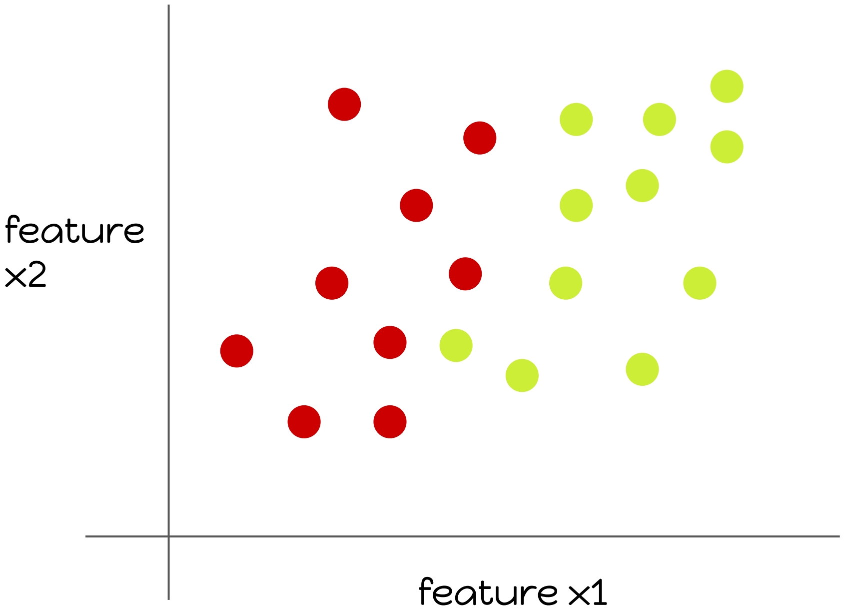
```
## load library
## install.packages("mlbench")
library(mlbench)
library(tidyverse)

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

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

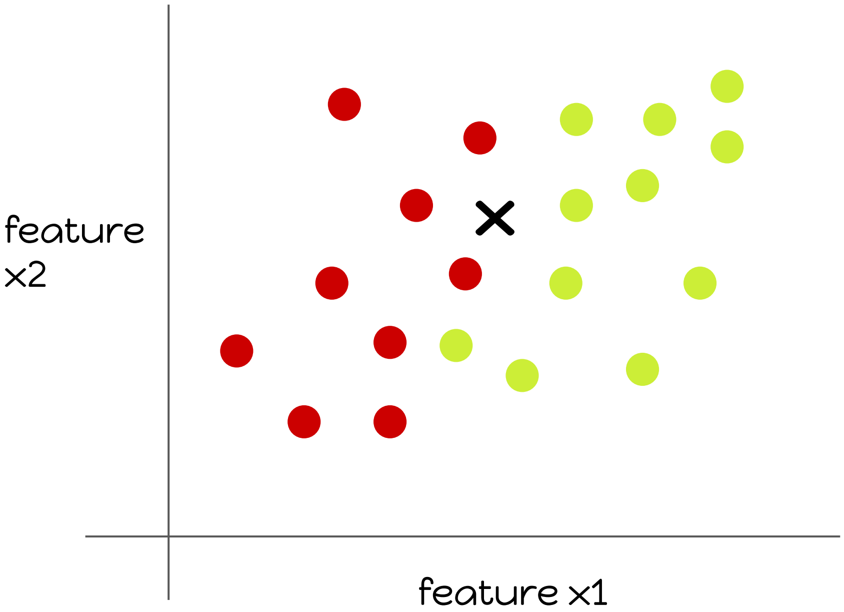
Caret Training Template

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

Model that we
want to train

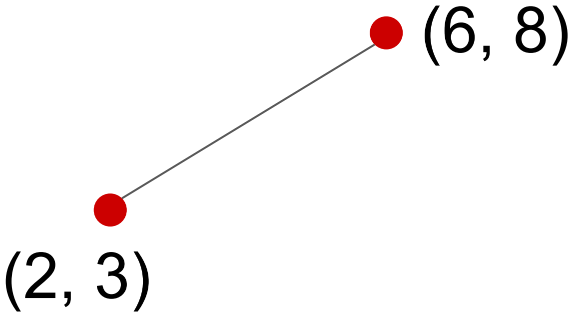
Our first machine





Euclidean Distance

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

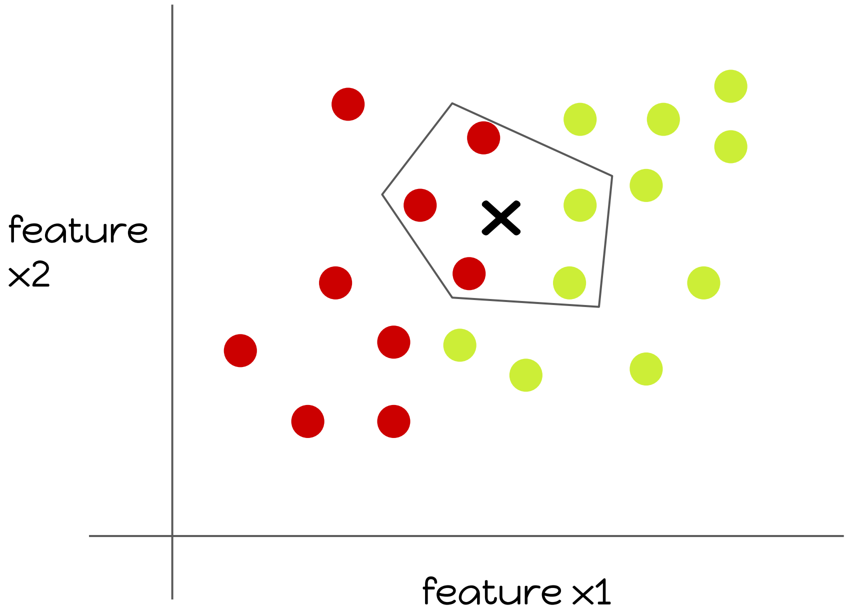


Euclidean Distance

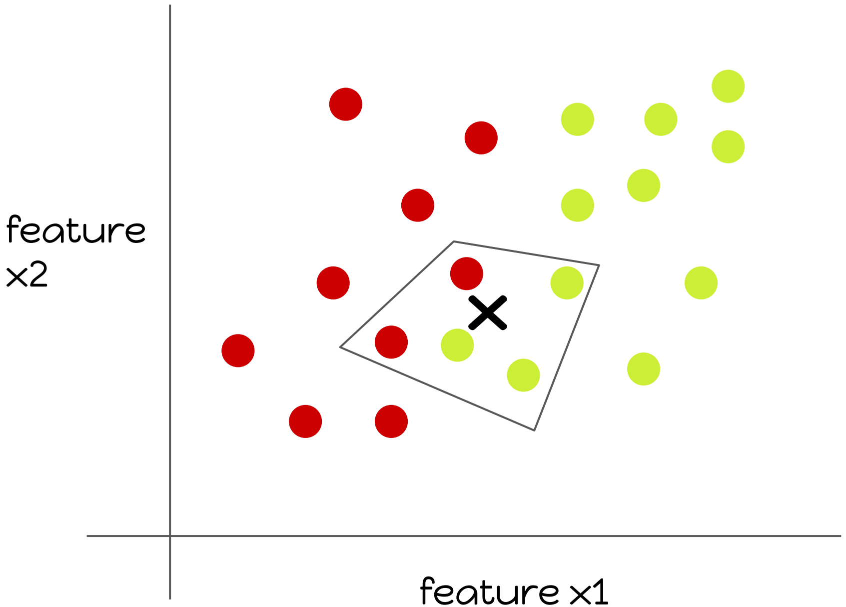
$$d = \sqrt{(x1 - x2)^2 + (y1 - y2)^2}$$

```
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%

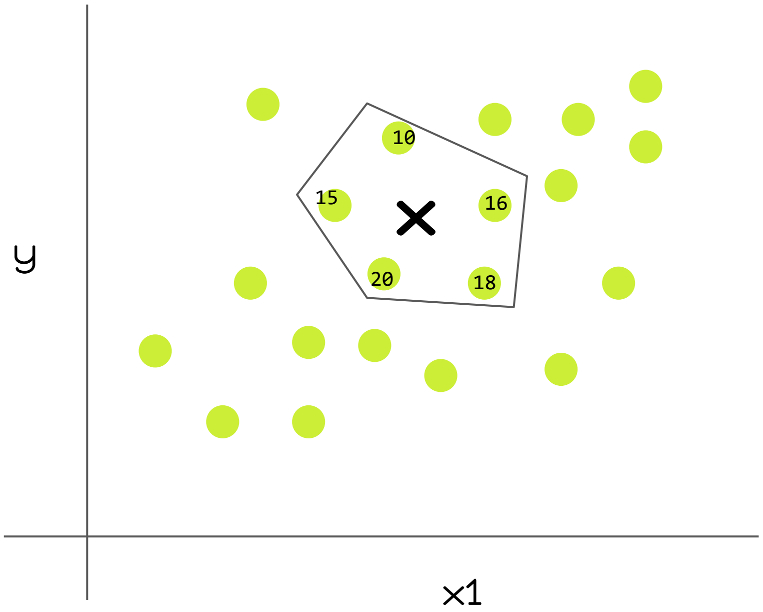
Majority Vote

Steps to train this algorithm

1. แต่งตั้ง สว 250 คน
2. ช่วยกันเลือกนายกฯ

เฮ้ย เด๋วๆๆๆ 55555555555+

We use average value for regression problem



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

Train test split (the easiest method)

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, ]
```

Very easy to train a machine in R

```
## 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))
```

5 Fold Cross Validation

Tuning

OFF/ON
AM
530 600 800 1000 1200 1400 1600 KHz
FM
88 92 96 100 104 106 108 MHz
VOLUME
TUNING
alamy

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

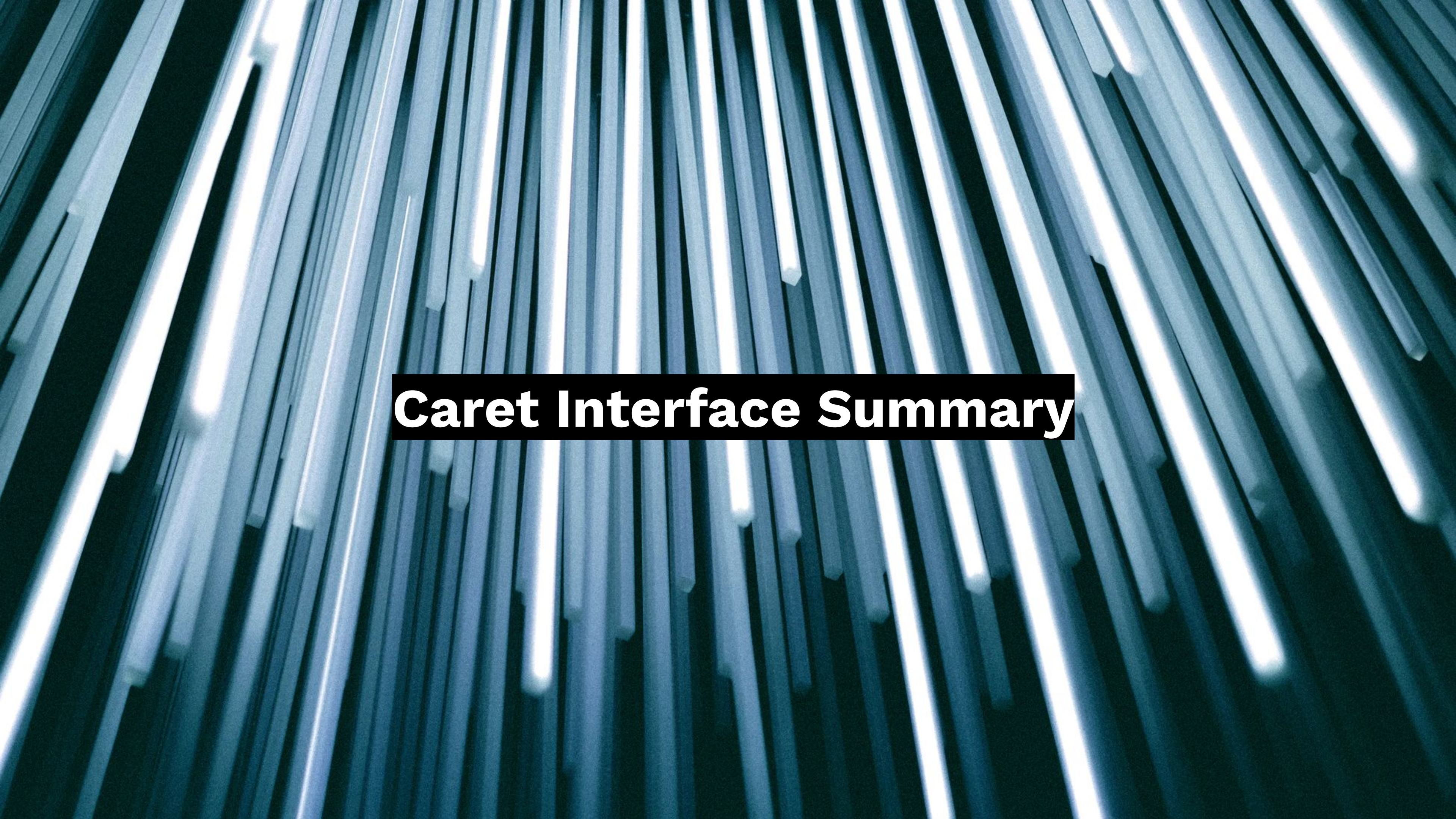
```
> knn_model
k-Nearest Neighbors

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.
```

k=5

Cross Validation
ช่วยเราเลือกค่า k ที่
ทำให้ 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



Caret Interface Summary

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

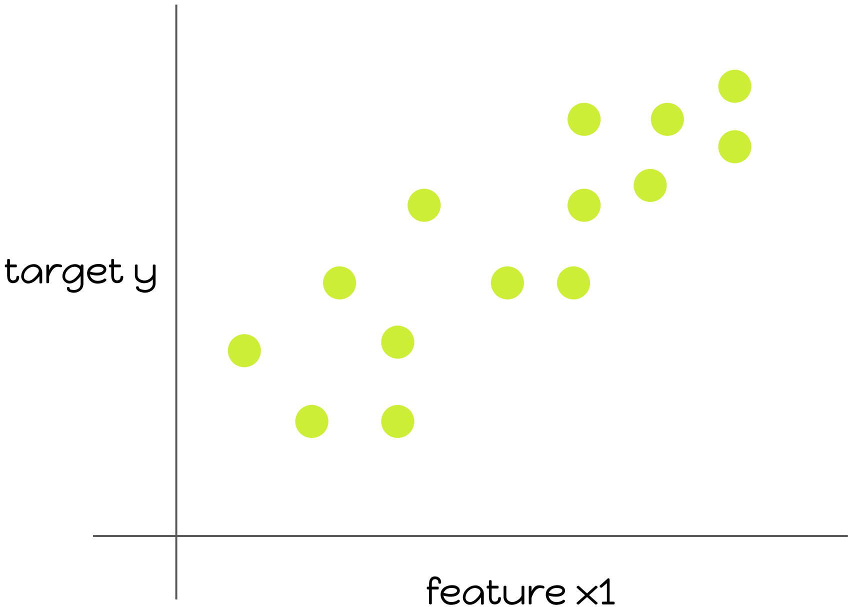
Classification - ROC Sens Specs

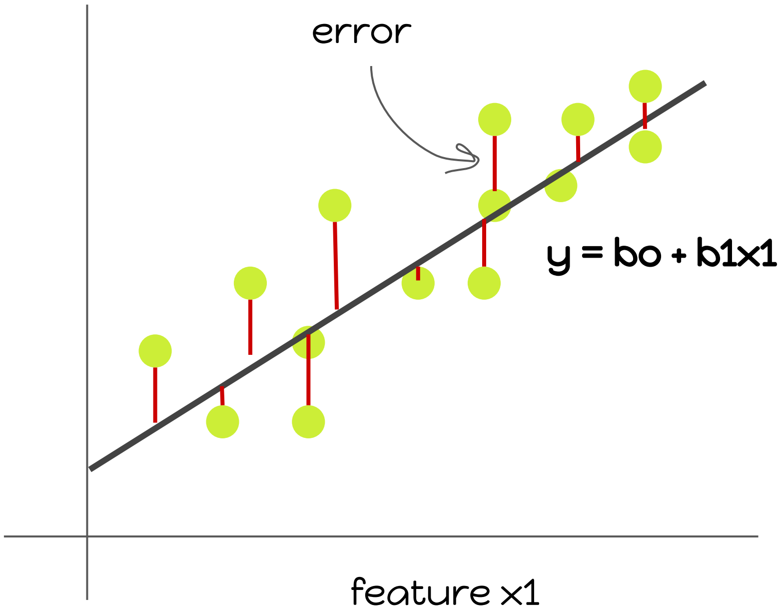
```
set.seed(42)

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
)
```

Linear Regression Explained





Linear Regression Model

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

prediction

y intercept

slope

Linear Regression finds
B0 and B1 that minimize
the error

Minimize Error

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

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

Sum of Squared Error
or RSS (for short)

Common Regression Metrics

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

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

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

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

Easy to compute in Spreadsheets

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

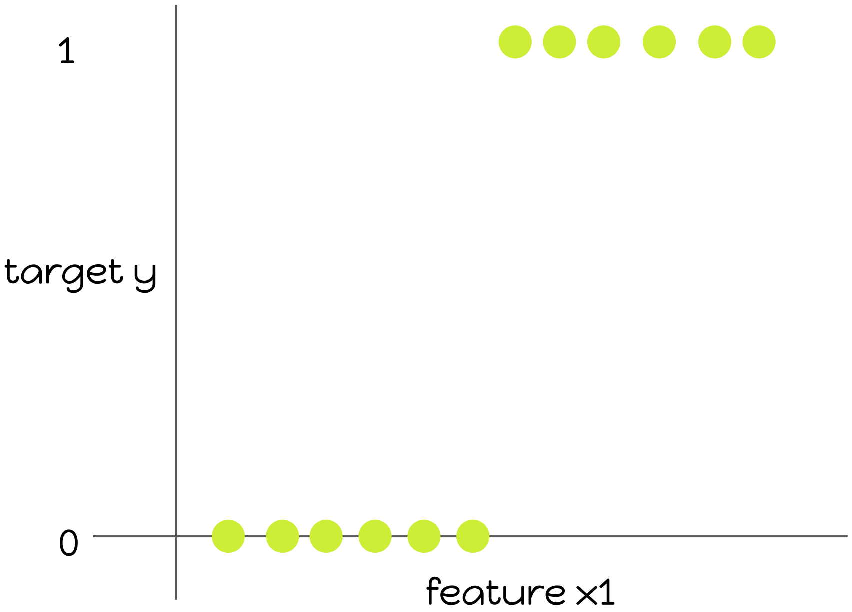
Build linear regression in R

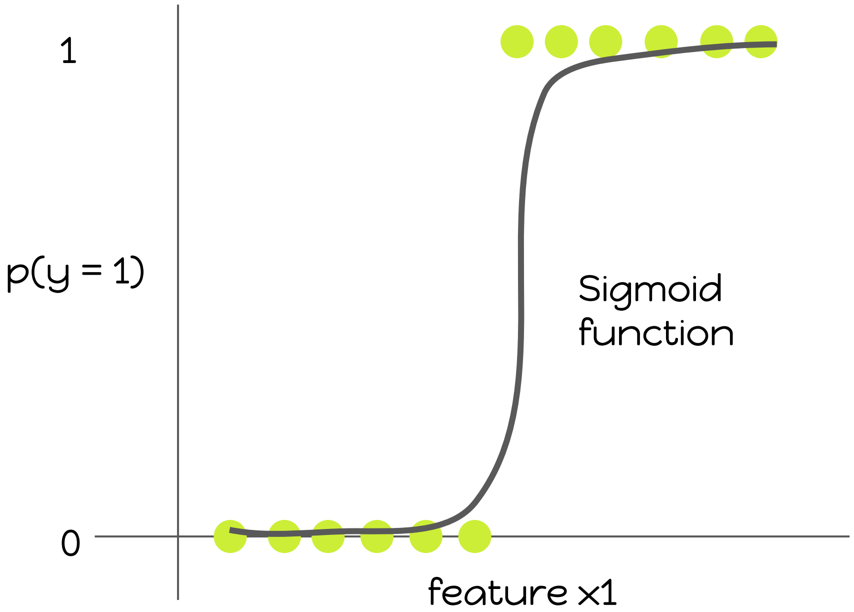
```
## train model with train_data
set.seed(99)
lm_model <- train(medv ~ rm + indus + crim,
                  data = train_data,
                  method = "lm")

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

Linear regression with K-Fold

```
## 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 for binary classification





Logistic is very similar to linear regression

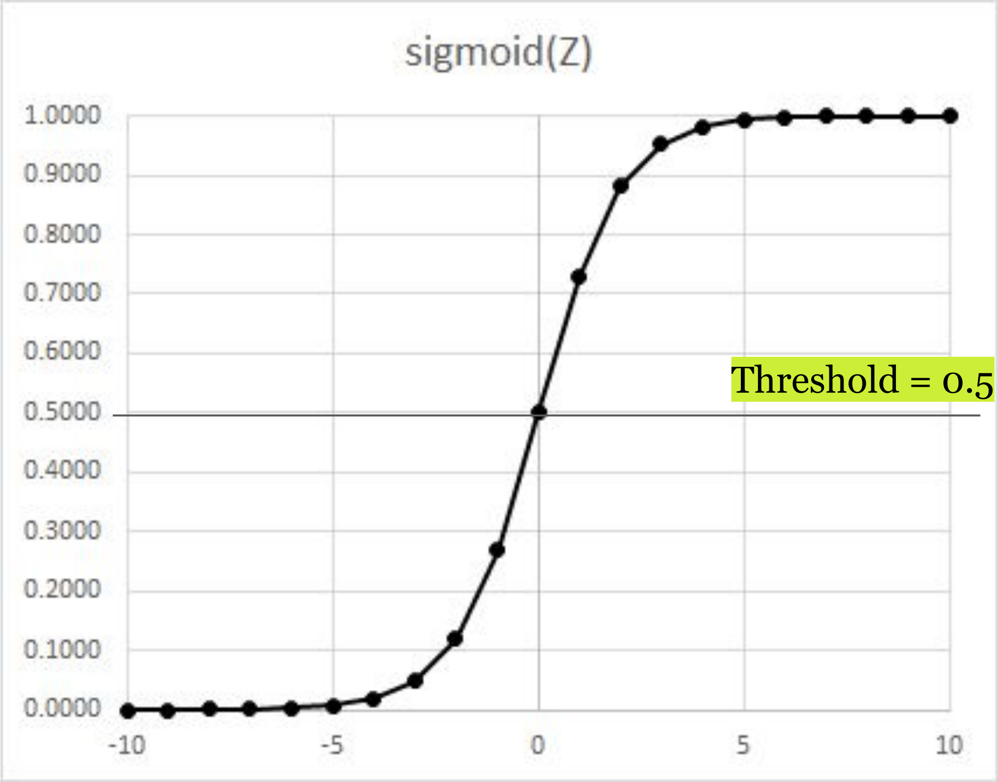
$$Z = B_0 + B_1X_1$$

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

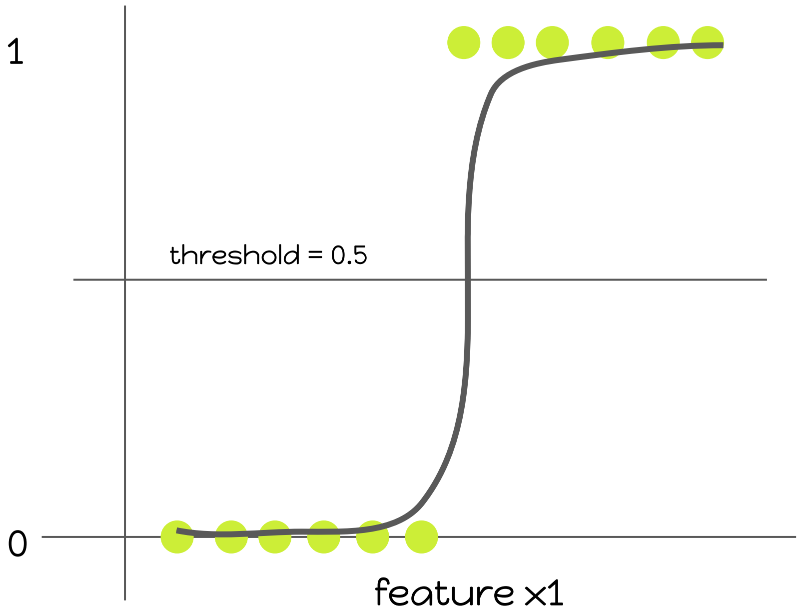
Sigmoid function

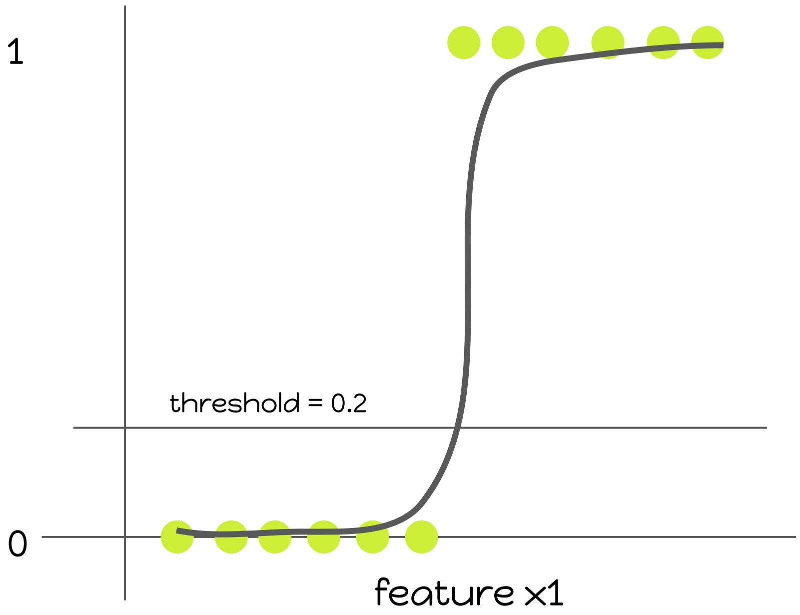
Z	sigmoid(Z)	y_hat
-10	0.0000	0
-9	0.0001	0
-8	0.0003	0
-7	0.0009	0
-6	0.0025	0
-5	0.0067	0
-4	0.0180	0
-3	0.0474	0
-2	0.1192	0
-1	0.2689	0
0	0.5000	0
1	0.7311	1
2	0.8808	1
3	0.9526	1
4	0.9820	1
5	0.9933	1
6	0.9975	1
7	0.9991	1
8	0.9997	1
9	0.9999	1
10	1.0000	1

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



Our prediction changes if threshold changes





R Code

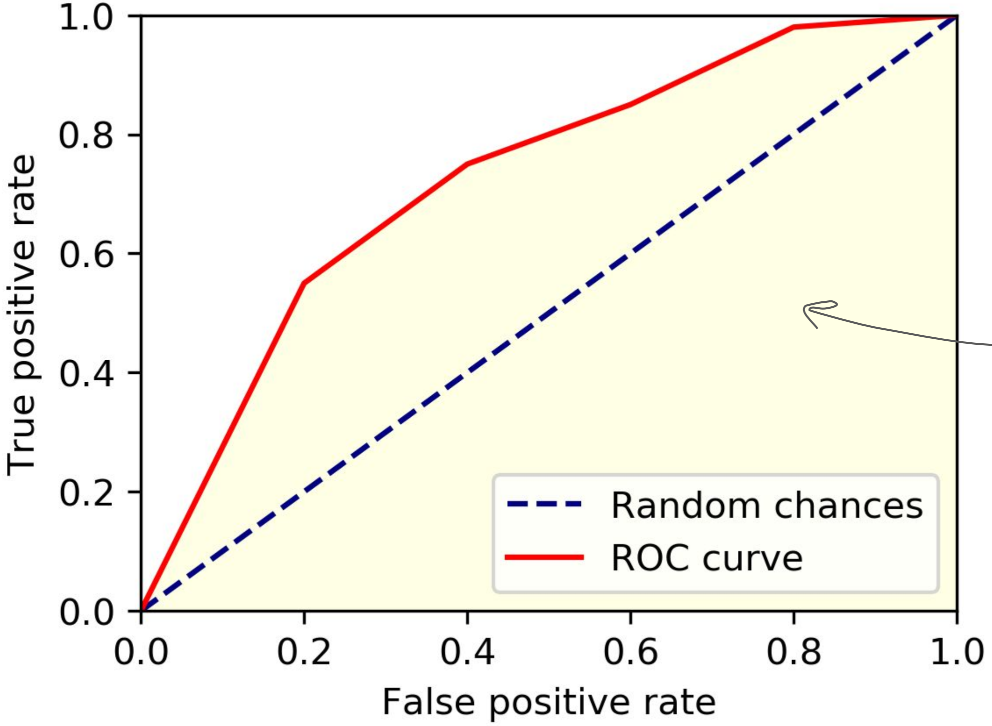
```
## train model with train_data
set.seed(99)
ctrl <- trainControl(method = “cv”, number = 5,
                     verboseIter = TRUE)

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)
```

Common classification metrics

- Accuracy
- Precision
- Recall
- F1

สามารถคำนวณได้
ง่ายๆจาก
Confusion Matrix

Common classification metrics



AUC

Area Under Curve

Interpretation in Thai

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

F1 Score

27.4.5 Balanced accuracy and F_1 score

Although we usually recommend studying both specificity and sensitivity, very often it is useful to have a one-number summary, for example for optimization purposes. One metric that is preferred over overall accuracy is the average of specificity and sensitivity, referred to as *balanced accuracy*. Because specificity and sensitivity are rates, it is more appropriate to compute the *harmonic* average. In fact, the F_1-*score*, a widely used one-number summary, is the harmonic average of precision and recall:

$$\frac{1}{\frac{1}{2}\left(\frac{1}{\text{recall}} + \frac{1}{\text{precision}}\right)}$$

Because it is easier to write, you often see this harmonic average rewritten as:

$$2 \times \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

when defining F_1.

[Chapter 27 Introduction to machine learning | Introduction to Data Science (rafalab.github.io)]

R Code - Confusion Matrix

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

Actual

Predicted

	neg	pos
neg	101 **TN**	33 **FN**
pos	14 **FP**	44 **TP**

```
## 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)
```

R **Regularized Regression**

1. Ridge Regression
2. Lasso Regression

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

R Lasso Regression (L1)

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

Normal RSS from Linear Regression

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

Lasso add this term to the error function

Ridge Regression (L2)

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

Normal RSS from Linear Regression

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

Ridge add this term to the error function

Regularization helps reduce overfitting

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

Lasso = RSS + 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

Ridge and Lasso in R

Machine Learning with caret in R | DataCamp

script.R

```r
# Train glmnet with custom trainControl and tuning: model
model <- train(
  y ~ .,
  data = overfit,
  tuneGrid = expand.grid(
    alpha = 0:1,
    lambda = seq(0.0001, 1, length=20)
  ),
  method = "glmnet",
  trControl = myControl
)

# Print model to console
model

# Print maximum ROC statistic
max(model$results$ROC)
```

ElasticNet model

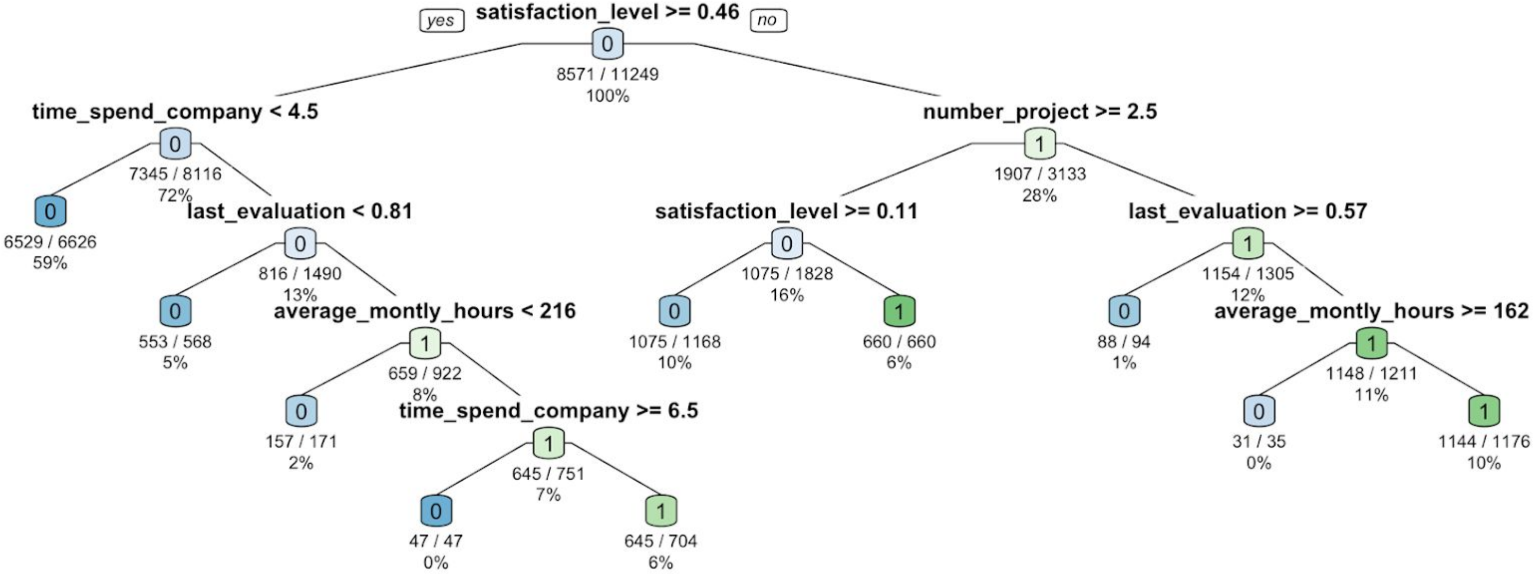
```
## train elasticnet model
set.seed(99)
ctrl <- trainControl(method = “cv”, number = 5,
                     verboseIter = TRUE)

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

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?

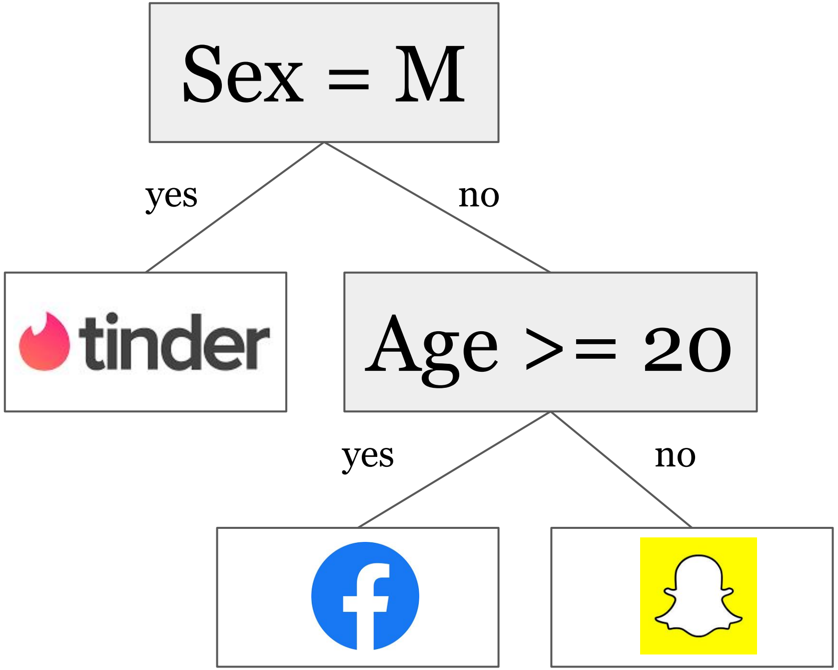
Age	Sex	App
15	M	tinder
20	M	tinder
16	F	
19	M	tinder
22	F	
20	F	

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 with K-Fold

```
## 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

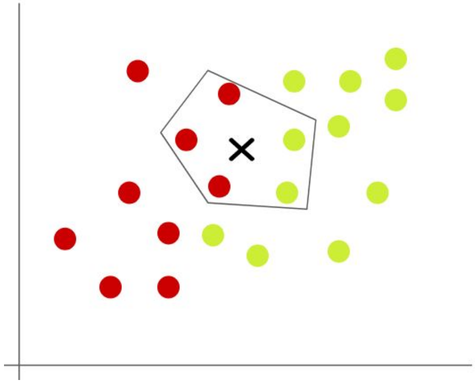
Parametric	**Non-Parametric**
Linear Regression	KNN
Logistic Regression	Decision Tree
Ridge Regression	Random Forest
Lasso Regression	

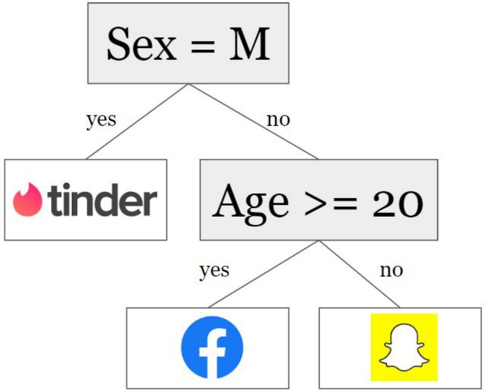
Regression is a Linear Combination (Parametric)

y_hat = b0 + b1x1 + b2x2 + b3x3 + b4x4

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)

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

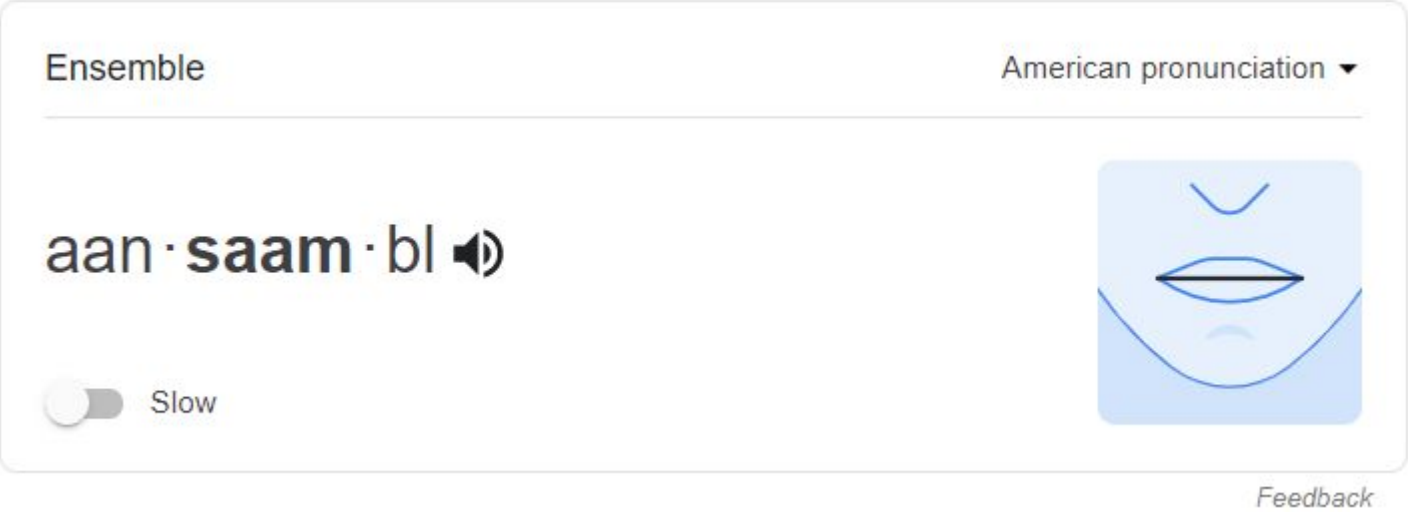
X2	X3	X4

Tree 100

Random Forest Code

```
## 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 Models



อาน ซาม เบิ้ล

นำโมเดลหลายๆตัวมาช่วยกันทำนายผล (Majority Vote)

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

Save our models for later use

```
saveRDS(model, “model.rds”)

model <- readRDS(“model.rds”)
```

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

Bootcamp Live 10
Introduction to Machine Learning

Website: https://datarockie.com