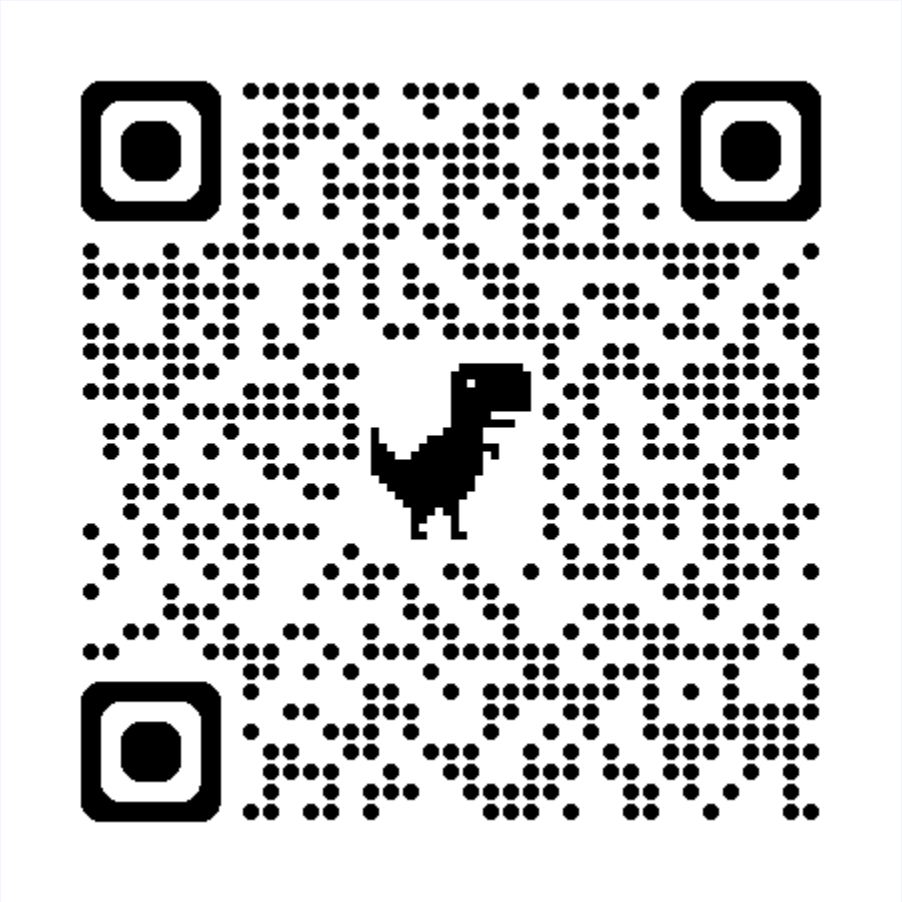
Machine learning with Python

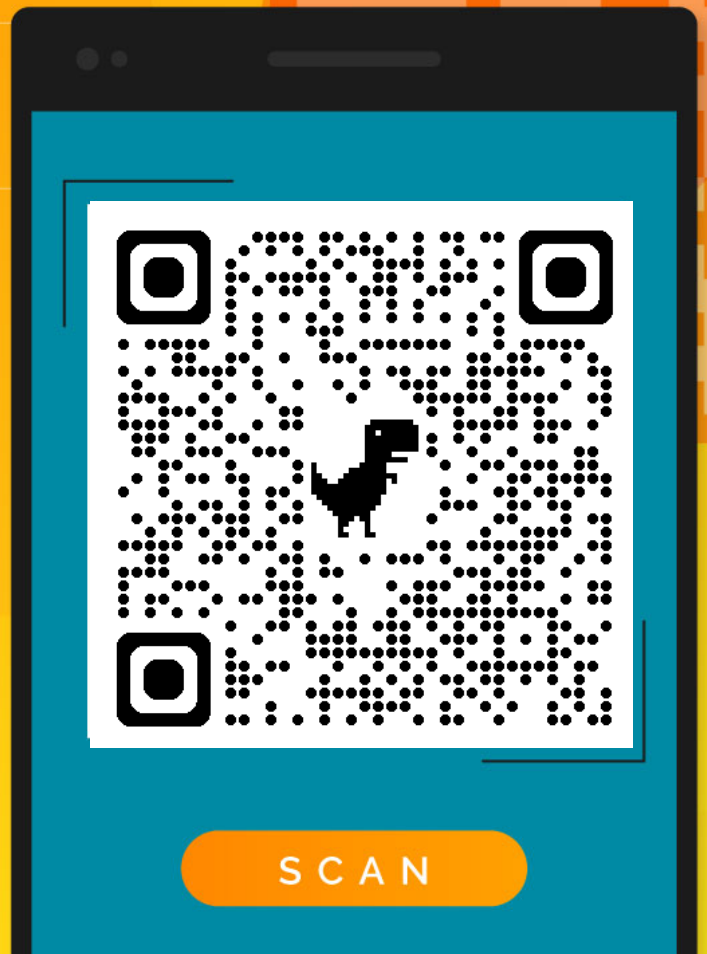
Lovnish Verma (lovnishverma.github.io)



https://github.com/Lovnishverma

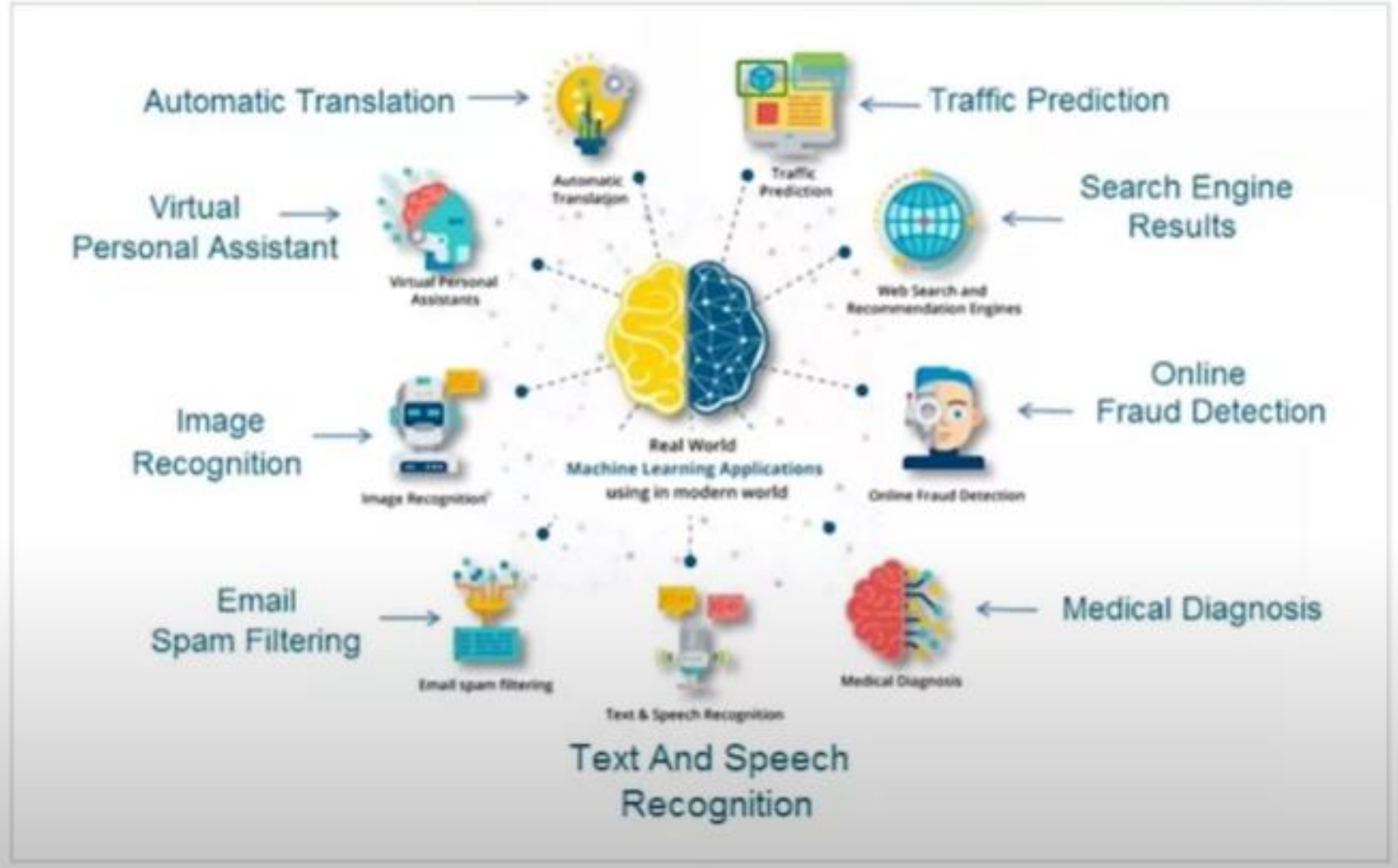
Stay Connected

LinkedIn®




1. ARTIFICIAL
INTELLIGENCE (AI)

Top Real-World Examples of Machine Learning



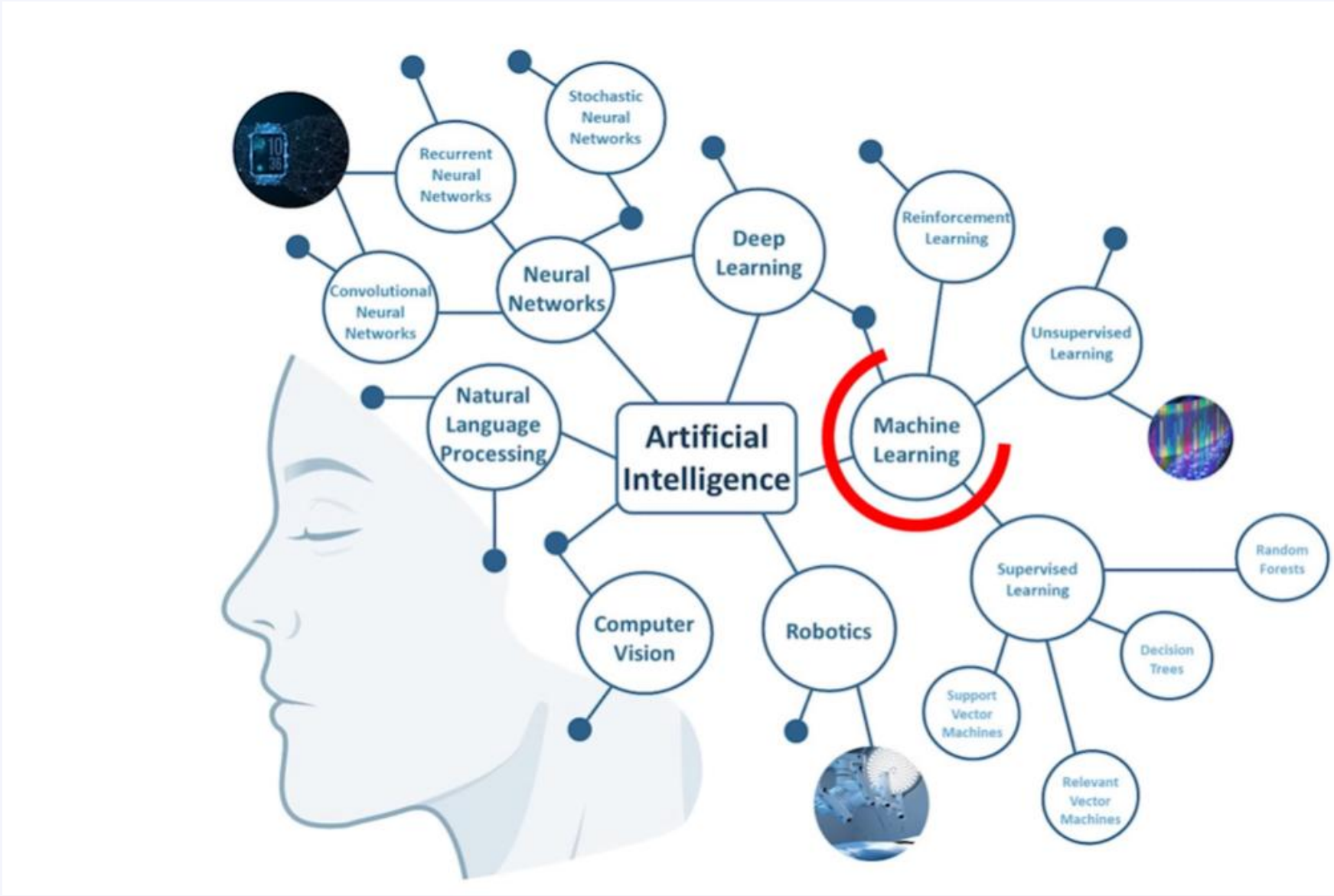
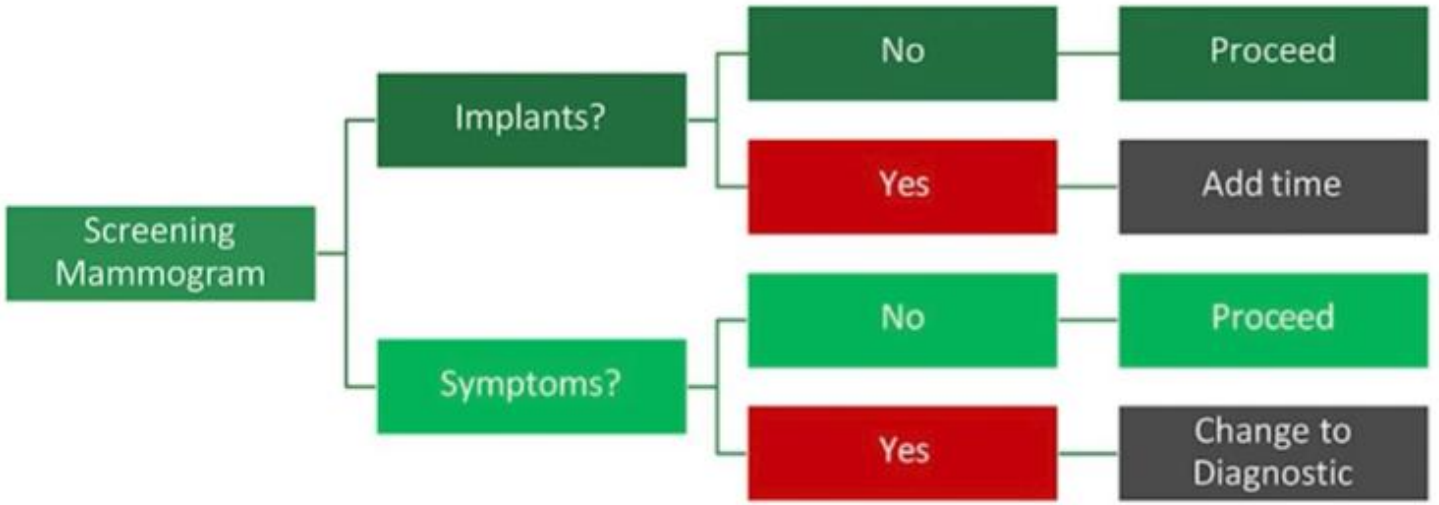
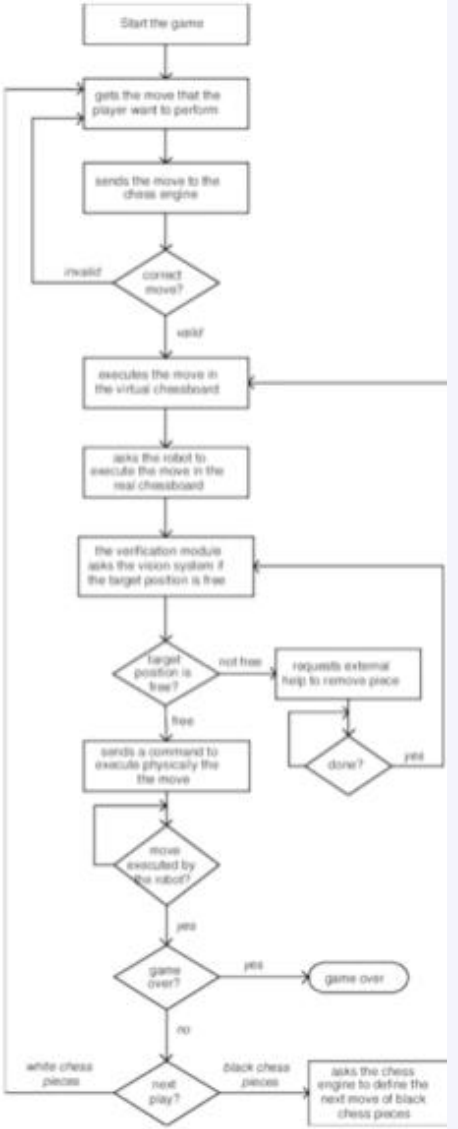
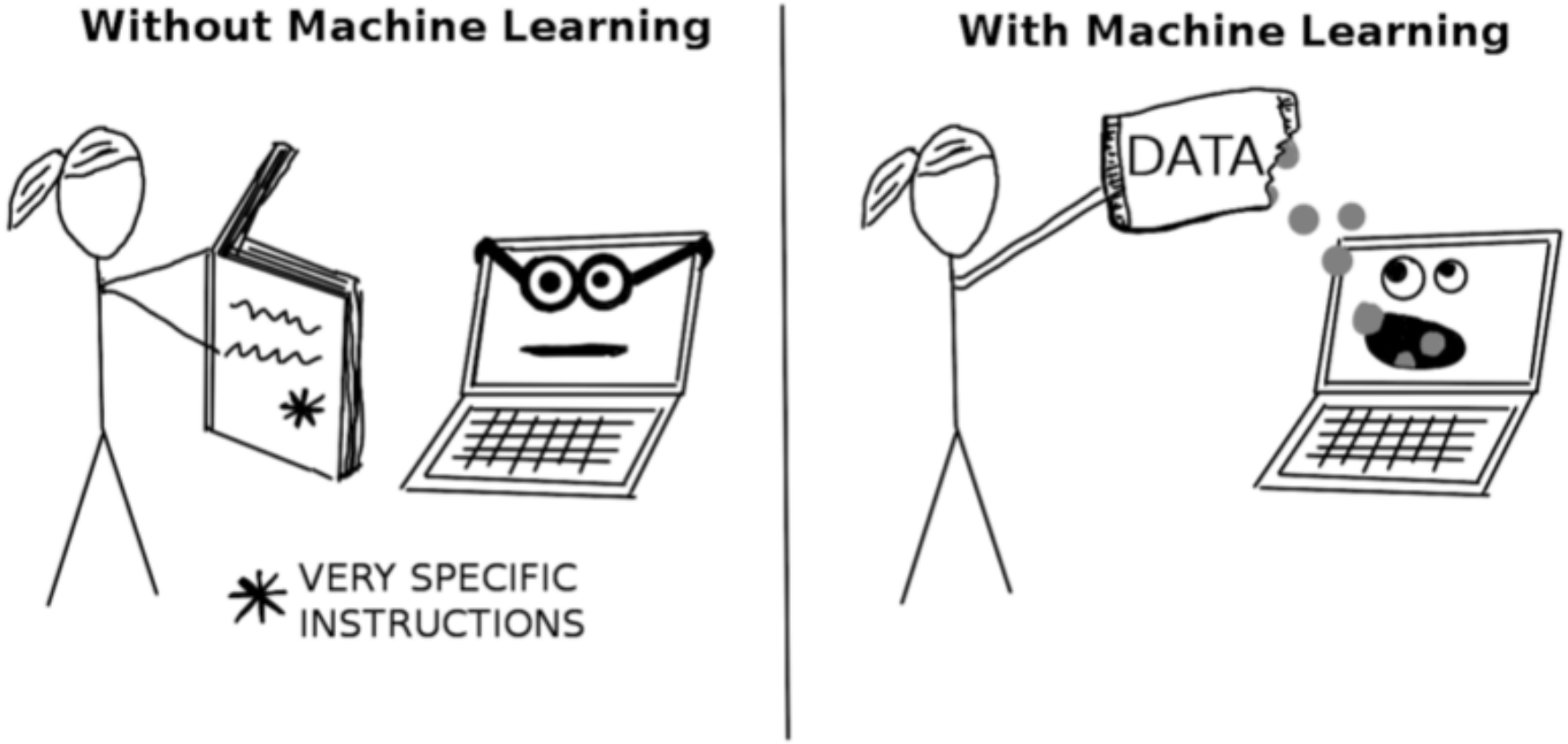
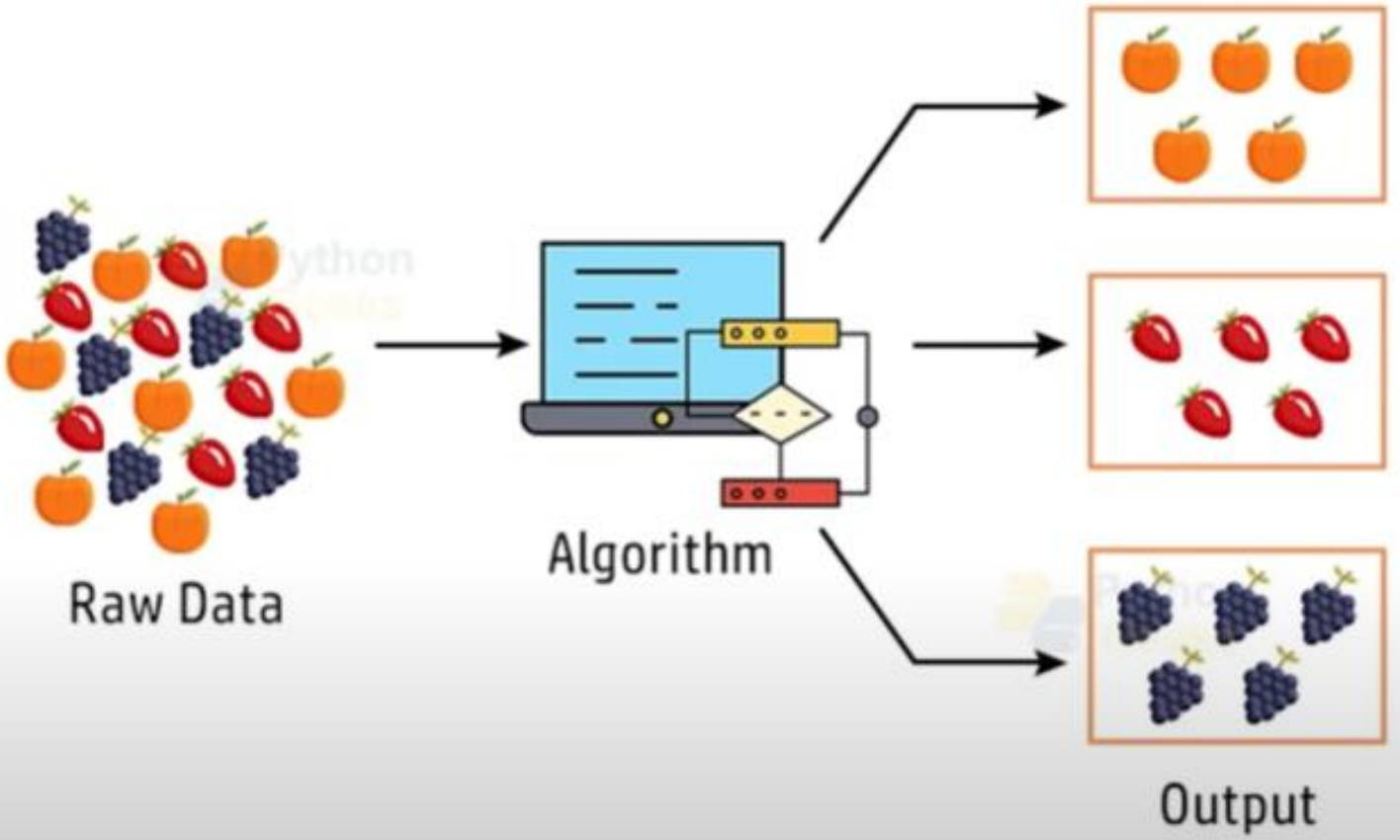

Stochastic Neural Networks
Recurrent Neural Networks
Convolutional Neural Networks
Neural Networks
Deep Learning
Reinforcement Learning
Unsupervised Learning
Natural Language Processing
Artificial Intelligence
Machine Learning
Supervised Learning
Random Forests
Decision Trees
Support Vector Machines
Relevant Vector Machines
Computer Vision
Robotics

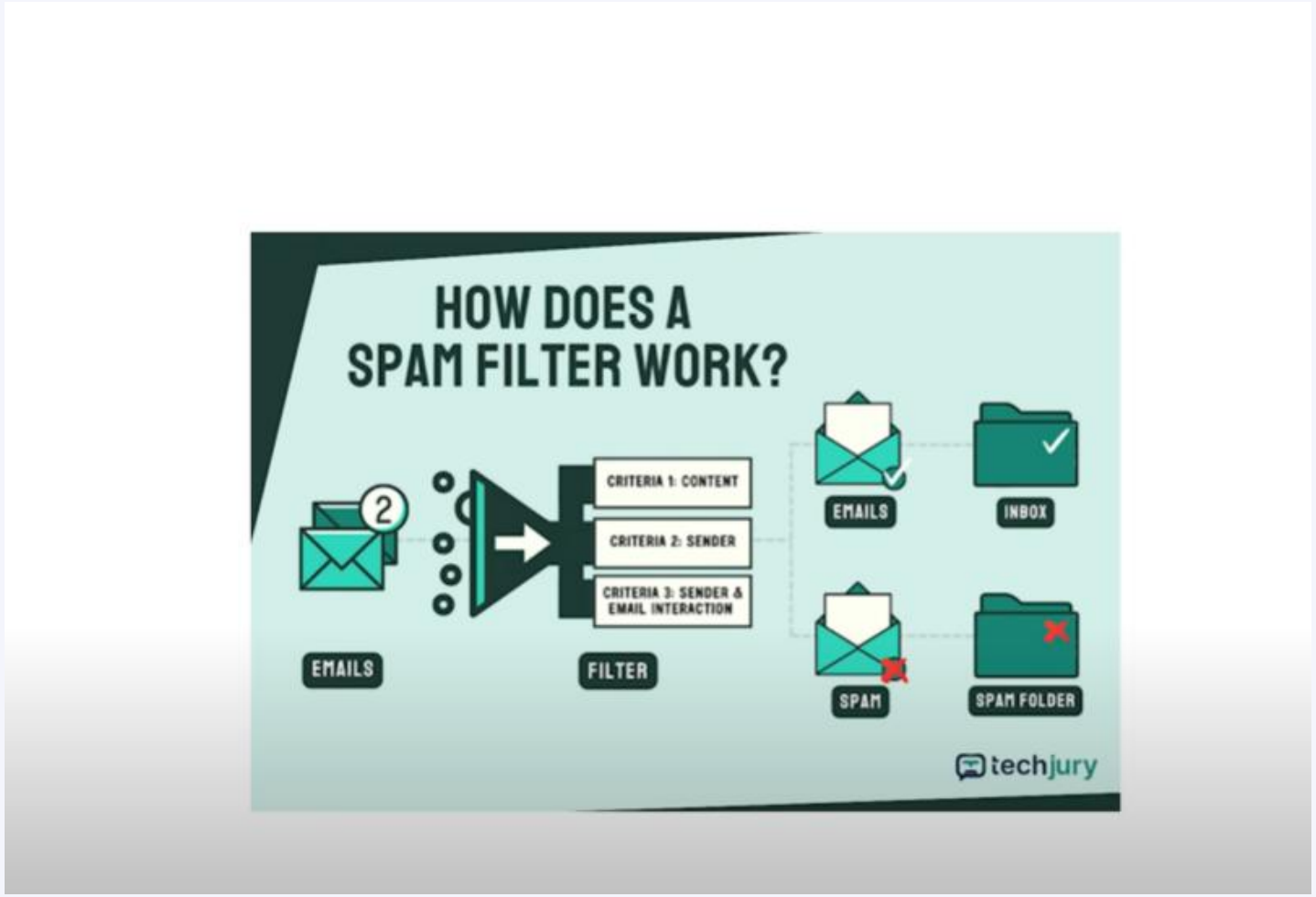
FIGURE 1. EXAMPLE OF RULES-BASED GUIDANCE FOR MAMMOGRAM SCREENINGS



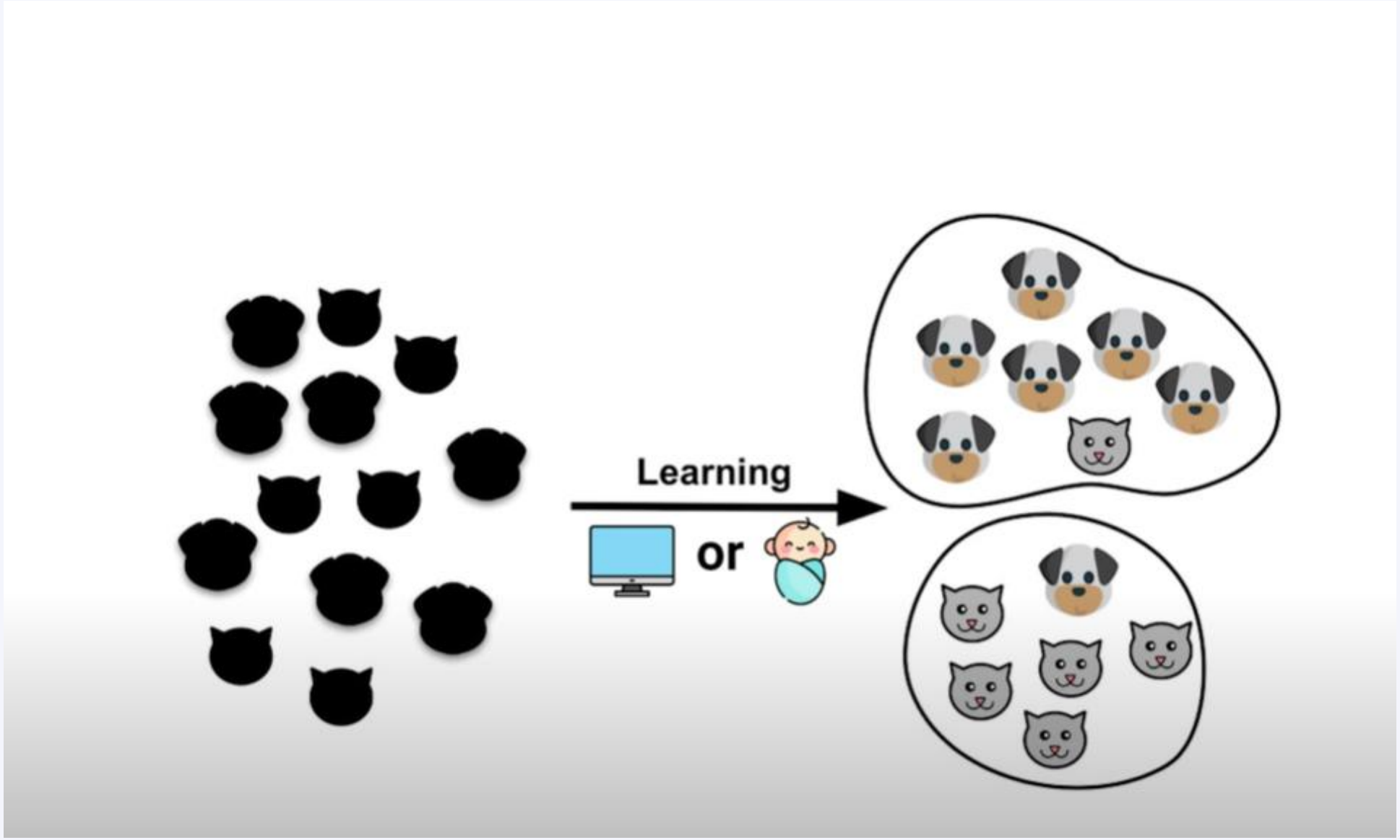



Without Machine Learning
With Machine Learning
DATA
VERY SPECIFIC
INSTRUCTIONS

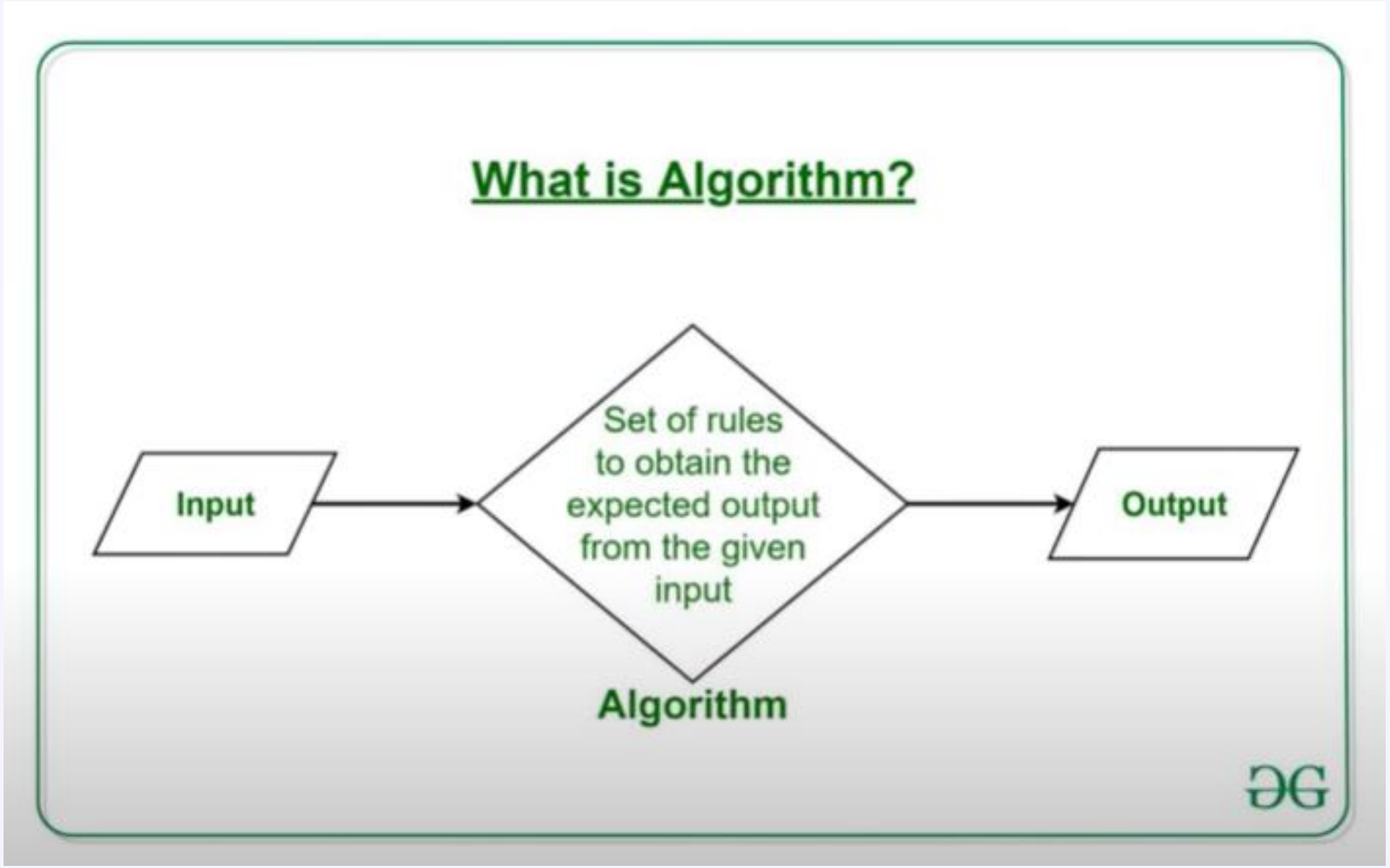

Raw Data
Algorithm
Output



https://github.com/lovnishverma/Python-Getting-Started/blob/main/Email_Spam_Detection_with_Machine_Learning.ipynb


Learning
or


3. ALGORITHM

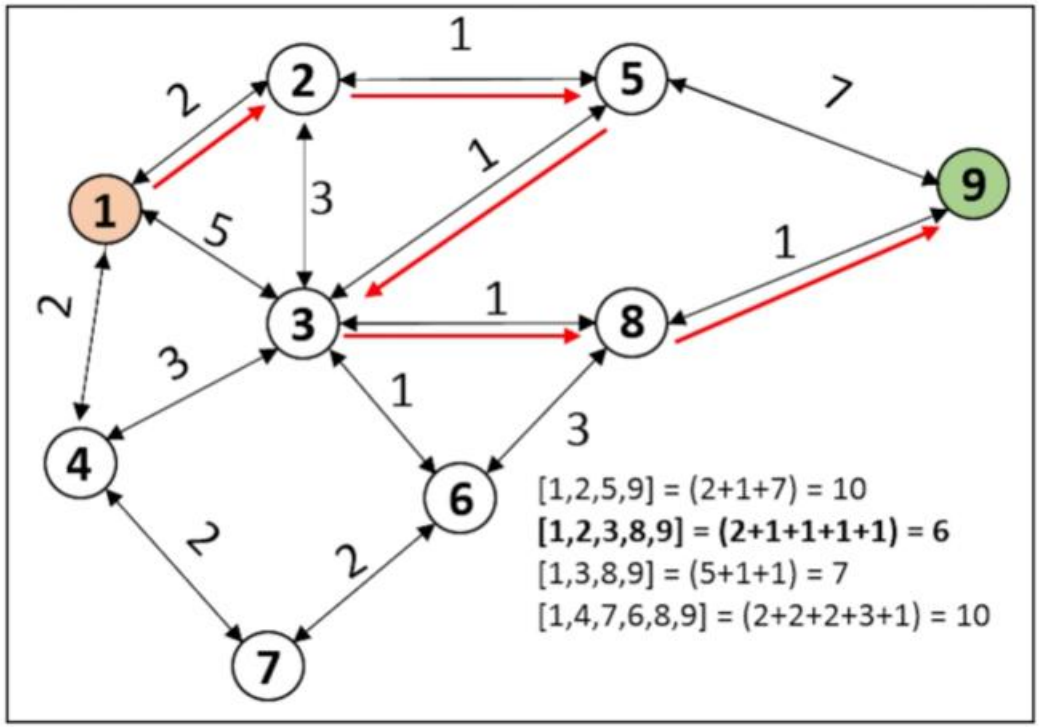

What is Algorithm?
Input
Set of rules to obtain the expected output from the given input
Output
Algorithm

ALGORITHM EXAMPLES

- Sorting
- Searching
- Encryption
- Data analysis
- much more…

SANDWICH MAKING ALGORITHM

1. Get two slices of bread
2. Spread butter on one side of each slice
3. Place a slice of cheese on one slice of bread
4. Add a slice of ham on top of the cheese
5. Put the other slice of bread on top, with the buttered side facing in



📘 **Key Concepts**

Works on **graphs** (nodes + weighted edges)

Finds **shortest path** from a **source** node to every other node

Uses a **priority queue** (min-heap) to always expand the node with the lowest tentative distance

Greedy approach: builds the shortest path incrementally

Dijkstra's Algorithm (Foundational)

Purpose: Finds the shortest path from a source to all other vertices in a weighted graph.

Usage: Google used it early on for shortest paths in small-scale networks.


4. DATA

- Numbers
- Text
- Images

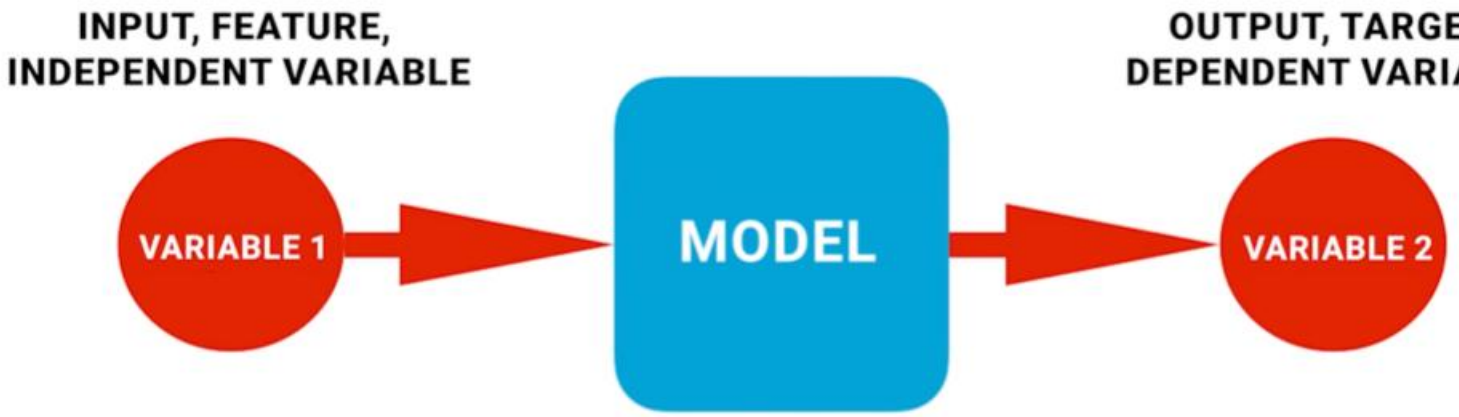
	A	B	C	D	E	F	G
1	OrderDa	Regio	Rep	Item	Unit	Unit Cos	Total
2	1/6/21	East	Jones	Pencil	95	1.99	189.05
3	1/23/21	Central	Kivell	Binder	50	19.99	999.50
4	2/9/21	Central	Jardine	Pencil	36	4.99	179.64
5	2/26/21	Central	Gill	Pen	27	19.99	539.73
6	3/15/21	West	Sorvino	Pencil	56	2.99	167.44
7	4/1/21	East	Jones	Binder	60	4.99	299.40
8	4/18/21	Central	Andrews	Pencil	75	1.99	149.25
9	5/5/21	Central	Jardine	Pencil	90	4.99	449.10
10	5/22/21	West	Thompso	Pencil	32	1.99	63.68
11	6/8/21	East	Jones	Binder	60	8.99	539.40
12	6/25/21	Central	Morgan	Pencil	90	4.99	449.10
13	7/12/21	East	Howard	Binder	29	1.99	57.71
14	7/29/21	East	Parent	Binder	81	19.99	1,619.19
15	8/15/21	East	Jones	Pencil	35	4.99	174.65
16	9/1/21	Central	Smith	Desk	2	125.00	250.00
17	9/18/21	East	Jones	Pen Set	16	15.99	255.84

City	Temperature (°C)		Relative Humidity (%)	
	Max.	Min.	Max.	Min.
Dammam	45.19	9.9	68.4	7.8
Abu Dhabi	43.02	14.9	78.6	15.9
Dubai	42.13	16.44	72.2	20.5
Kuwait	46.9	7.9	69.8	5.05
Doha	42.02	15.24	73.3	15.3
Bahrain	39.7	15.7	69.4	23.7
Muscat	37.8	17.8	87.3	30.4

R: 95 G: 26 B: 73	R: 129 G: 49 B: 88	R: 134 G: 69 B: 99
R: 109 G: 32 B: 74	R: 129 G: 47 B: 87	R: 154 G: 76 B: 102
R: 112 G: 45 B: 96	R: 133 G: 57 B: 96	R: 170 G: 89 B: 106

	A	B	C	D	E	F	G	H	I	J	K
1	**ALL QUERIES**	**count**		**LOCATIONS**			**BRANDS**			**CONCEPTS**	
2	berlin	3277		berlin	3277		google	627		data	305
3	straße	845		germany	704		twitter	303		information	177
4	design	747		new york	668		indesign	292		time	161
5	new	707		weimar	608		illustrator	183		art	159
6	germany	704		toronto	589		bloomberg	155		world	145
7	new york	668		leipzig	378		wordpress	146		open	125
8	google	627		london	282		windows	142		free	125
9	weimar	608		deutschland	195		chrome	141		one	110
10	toronto	589		oxford	166		mac	134		color	109
11	magazine	400		usa	165		facebook	127		money	104
12	oderberger	389		canada	144		iphone	120		nicht	93
13	leipzig	378		frankfurt	136		android	118		change	87
14	map	348		eisenberg	132		kindle	117		zeit	82
15	aktueller standort	334		hamburg	130		samsung	113		ich	69
16	data	305		ontario	126		amazon	100		life	64
17	twitter	303		brooklyn	121		adobe	77		old	64
18	euro	297		lisbon	105		ipod	65		space	58
19	indesign	292		jena	82		wikipedia	57		home	55
20	london	282		cuba	80		youtube	53		white	54
21	ave	276		portugal	72		gmail	52		death	53
22	der	269		kanada	71		tilemill	52		future	52
23	online	261		vereinigte staate	71		gestalten	51		two	50
24	street	230		greece	51		ipad	47		war	50
25	graphic	217		washington	51		sdw	45		system	44
26	525166667134	215		israel	48		nexus	44		number	43
27	app	207		sheffield	47		sims	43		growth	42
28	css	197		switzerland	47		leaflet	41		gold	39


5. MODEL



EX: LINEAR REGRESSION:

$$Y_i = \beta_0 + \beta_1 X_i$$

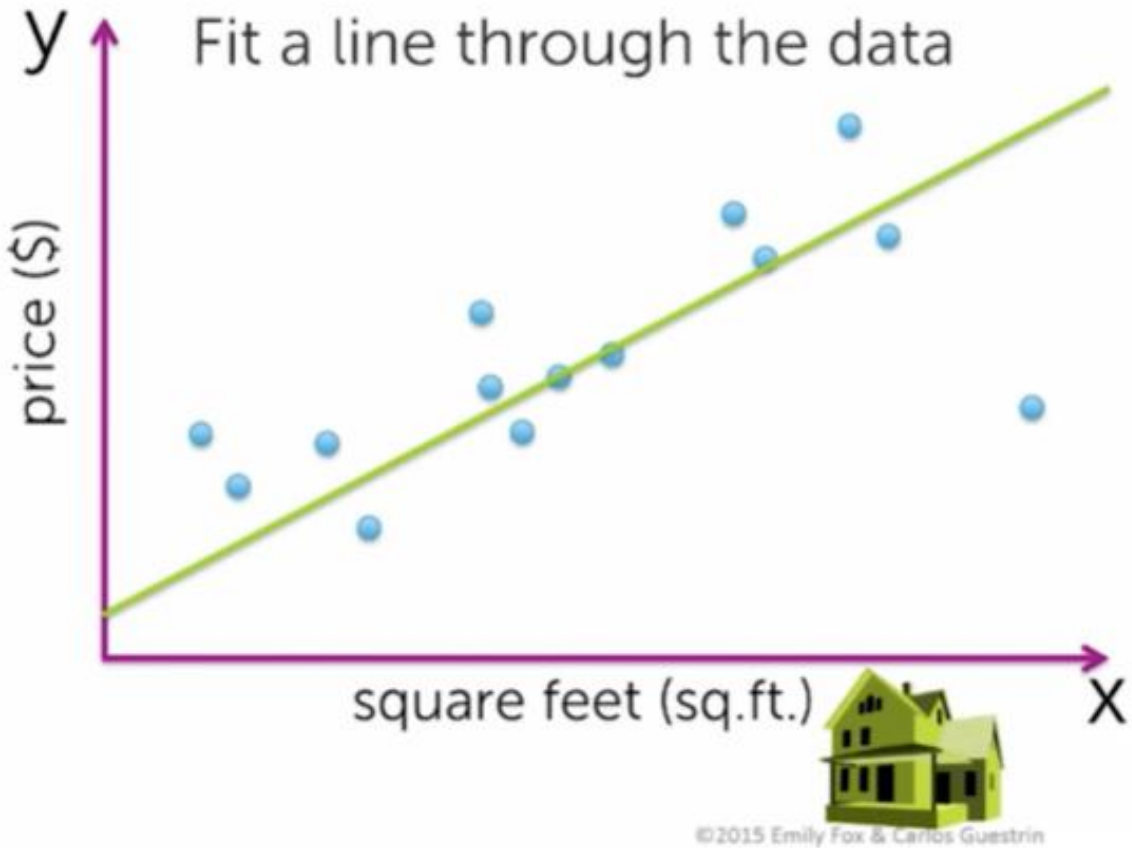
Constant/Intercept

Independent Variable

Dependent Variable

Slope/Coefficient

Use a **linear** regression model



13


6. MODEL FITTING

🔄 Process of Model Fitting

Choose a model

e.g., Linear Regression, Decision Tree, Neural Network, etc.

Split data

Training data: Used to fit the model

Testing/Validation data: Used to evaluate performance

Train/Fit the model

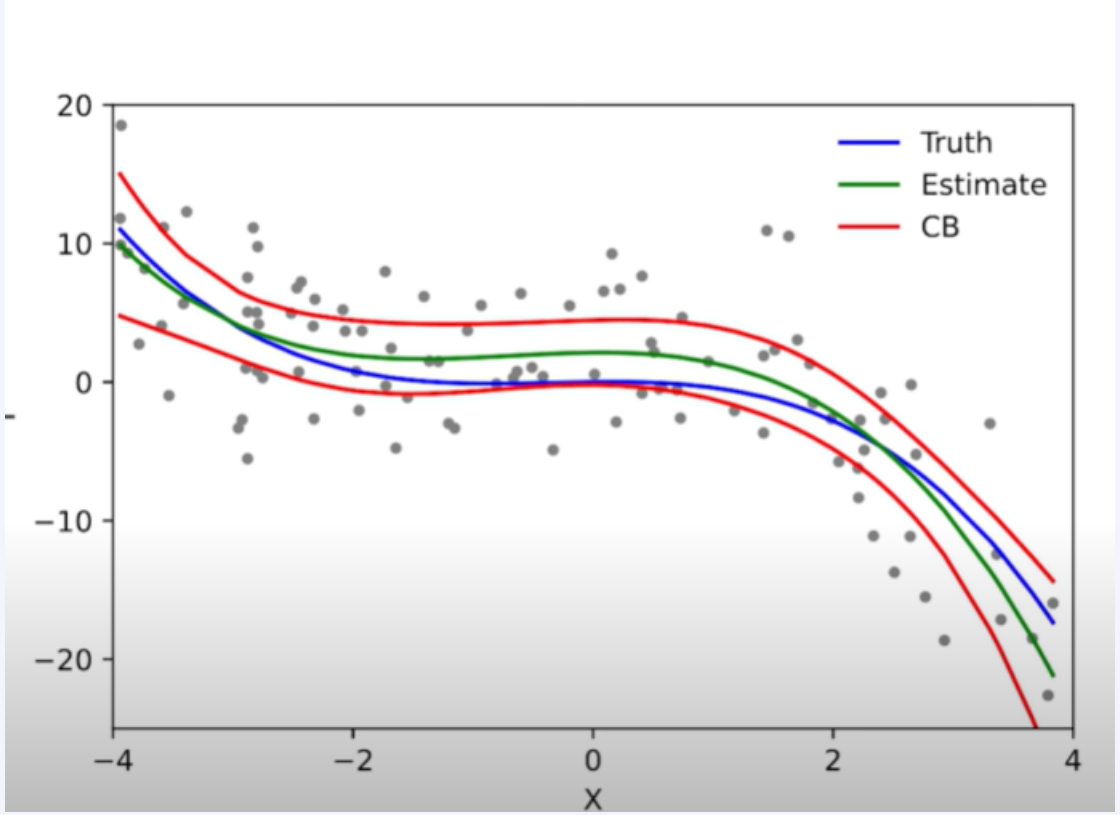
Use training data to learn model parameters (e.g., weights in linear regression)

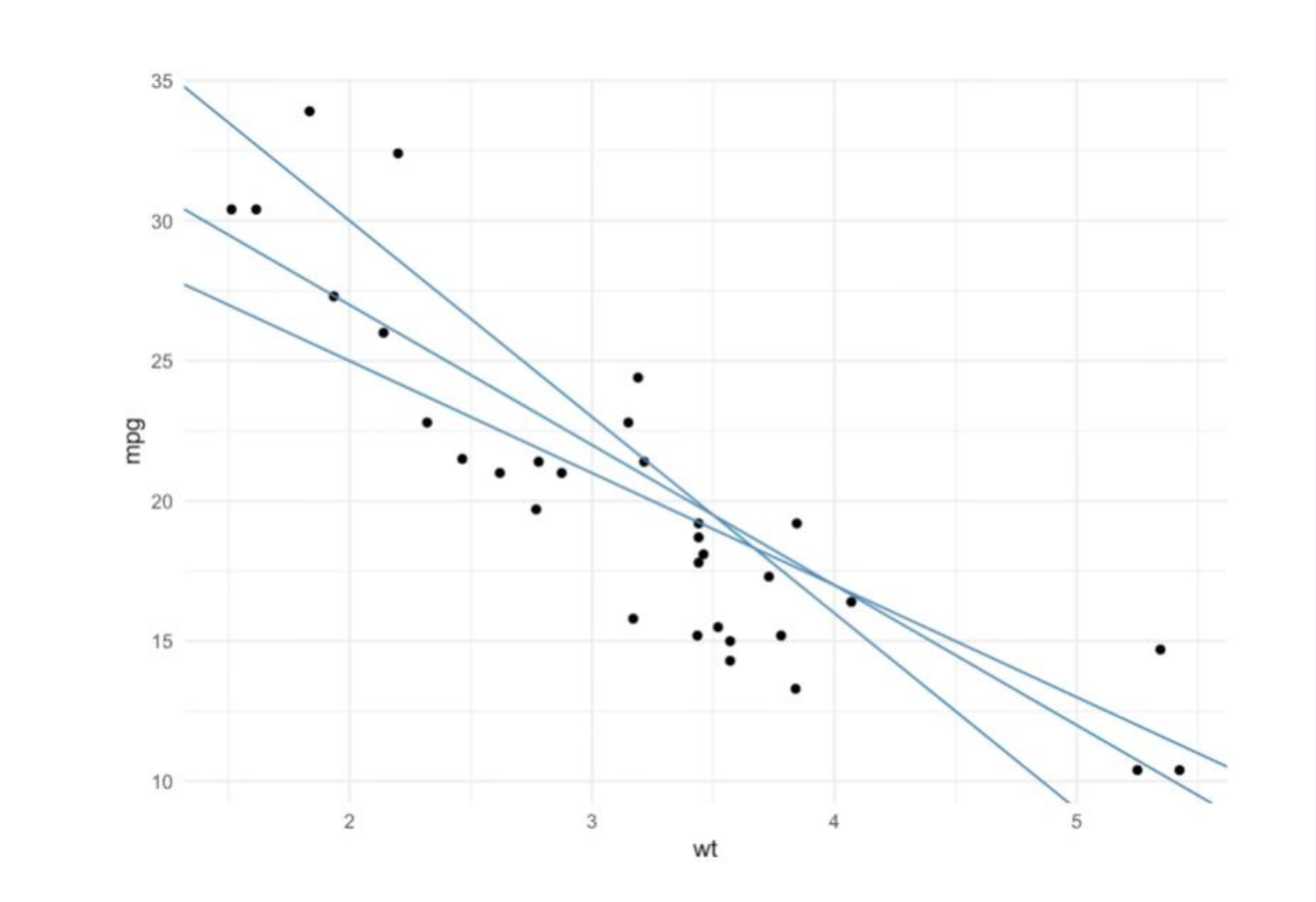
Evaluate

Use metrics like accuracy, R^2, RMSE, etc., to measure how well the model fits

Tune/Improve

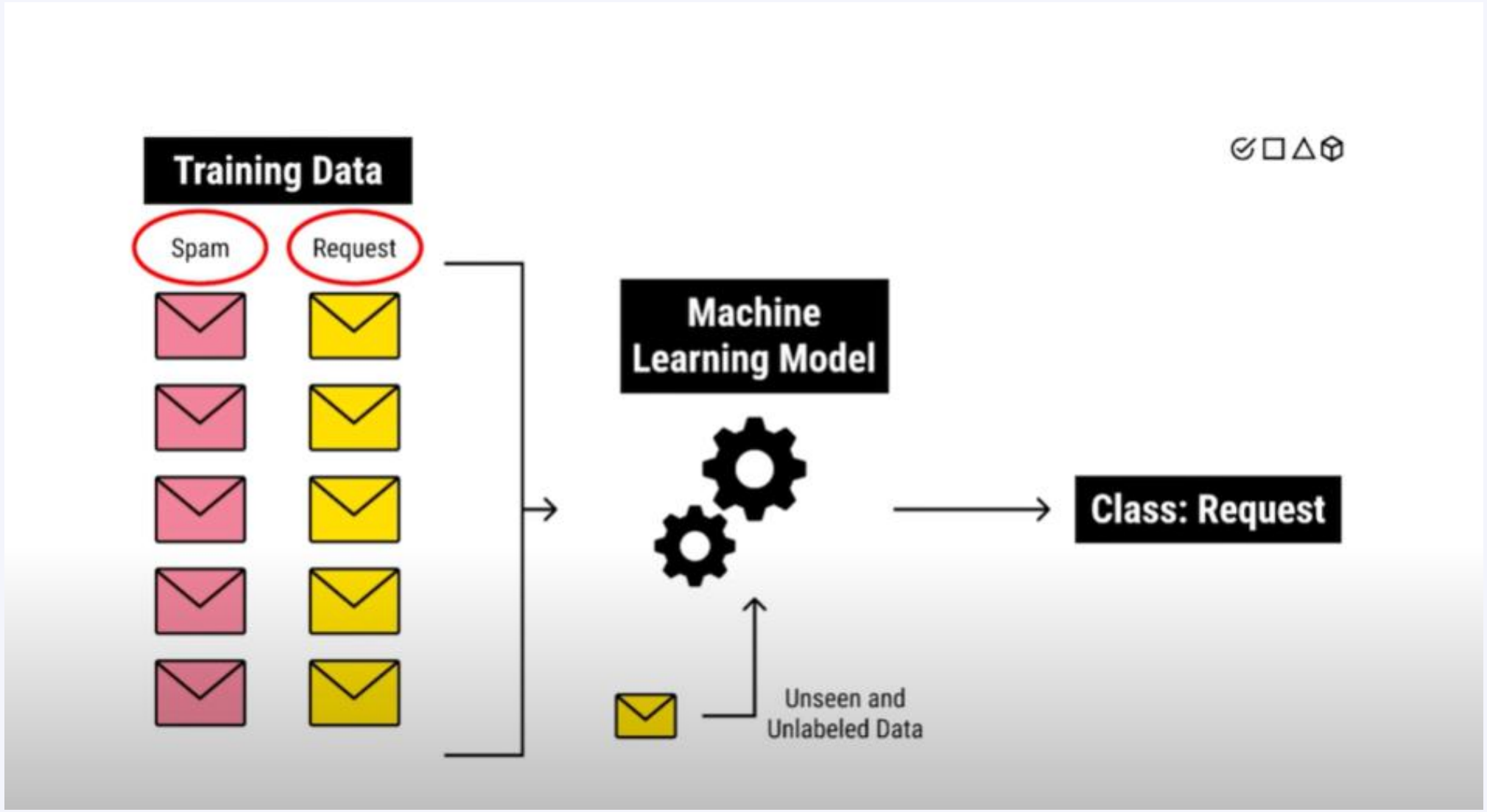
Adjust hyperparameters, use cross-validation, feature engineering, etc.

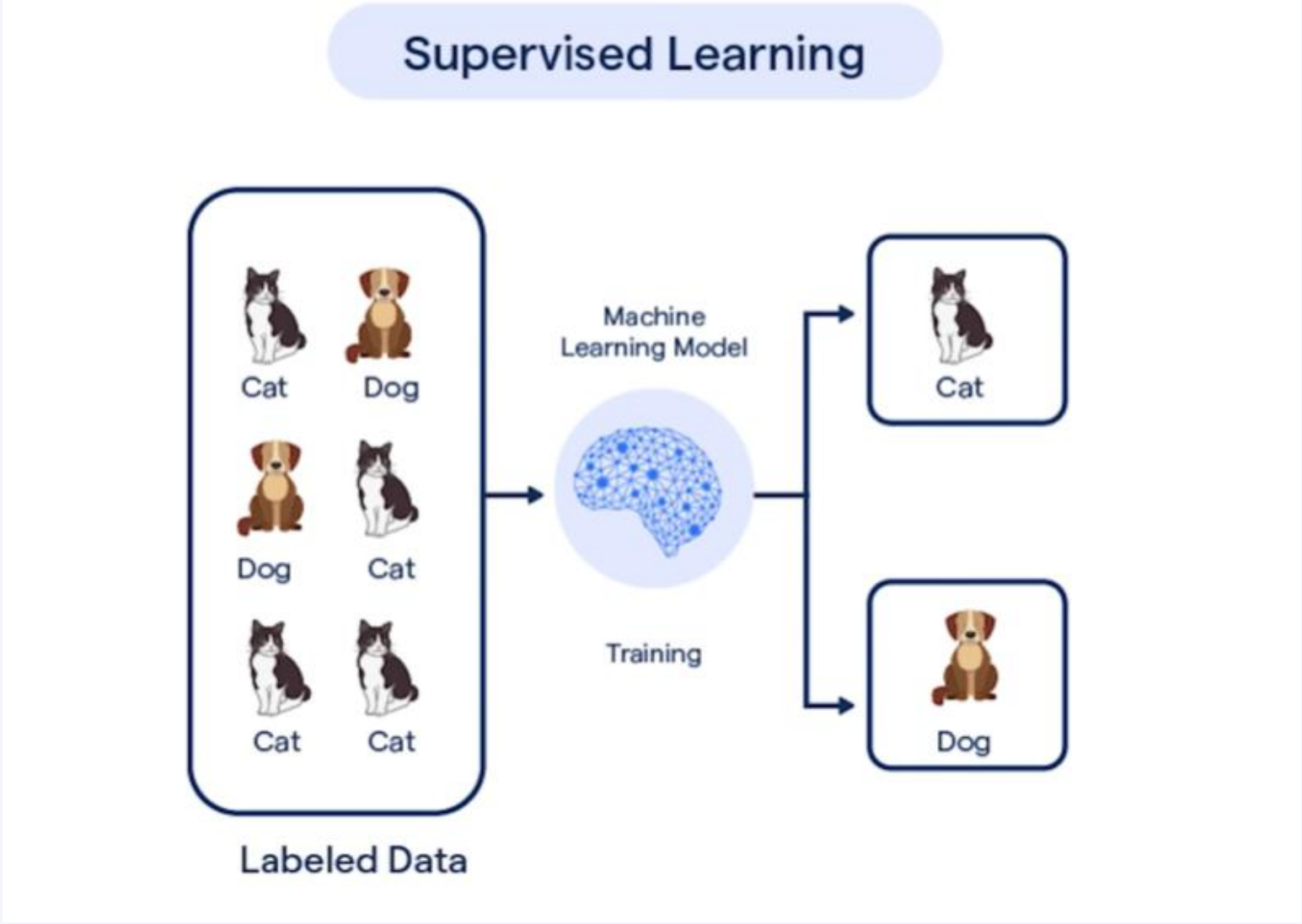




Model Fitting in Linear Regression


7. TRAINING DATA

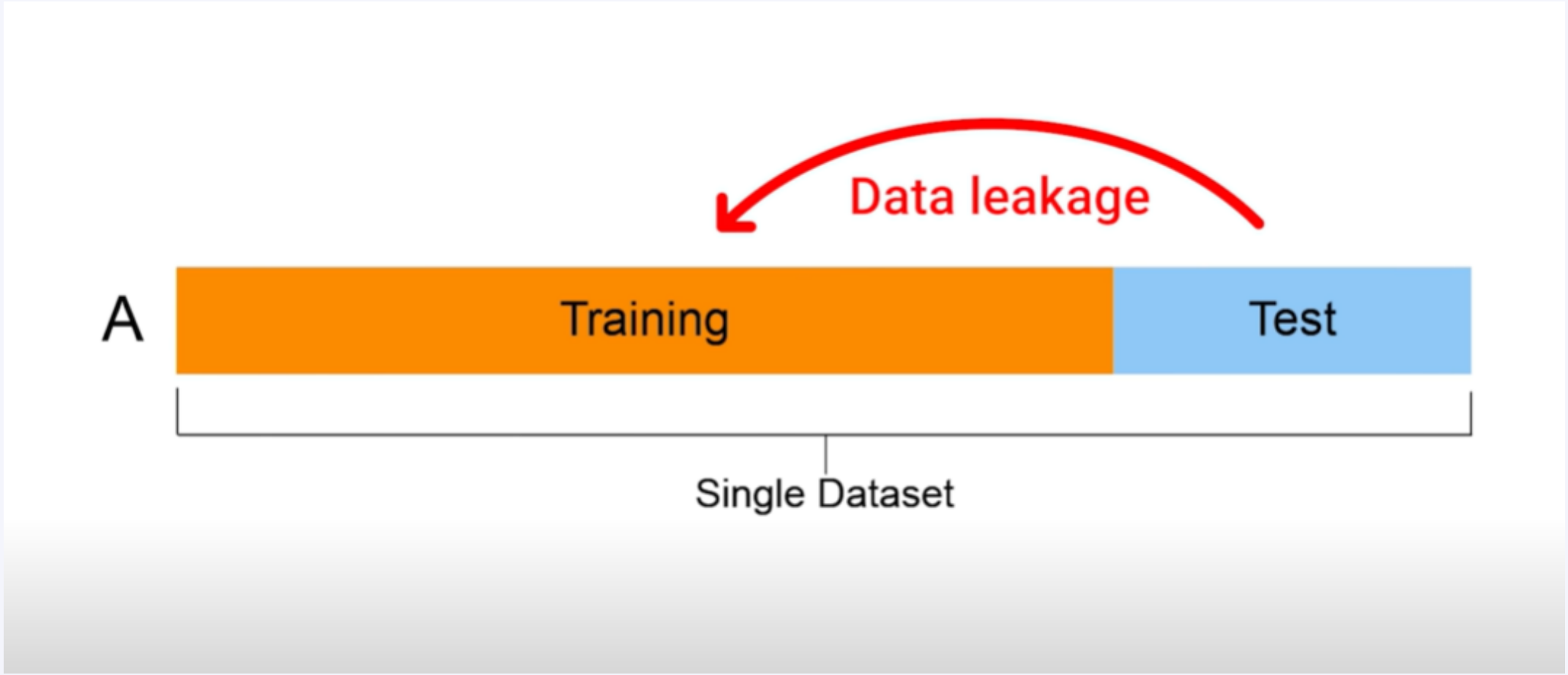

Training Data
Spam
Request
Machine Learning Model
Class: Request
Unseen and Unlabeled Data


Supervised Learning
Cat
Dog
Dog
Cat
Cat
Cat
Labeled Data
Machine
Learning Model
Training
Cat
Dog

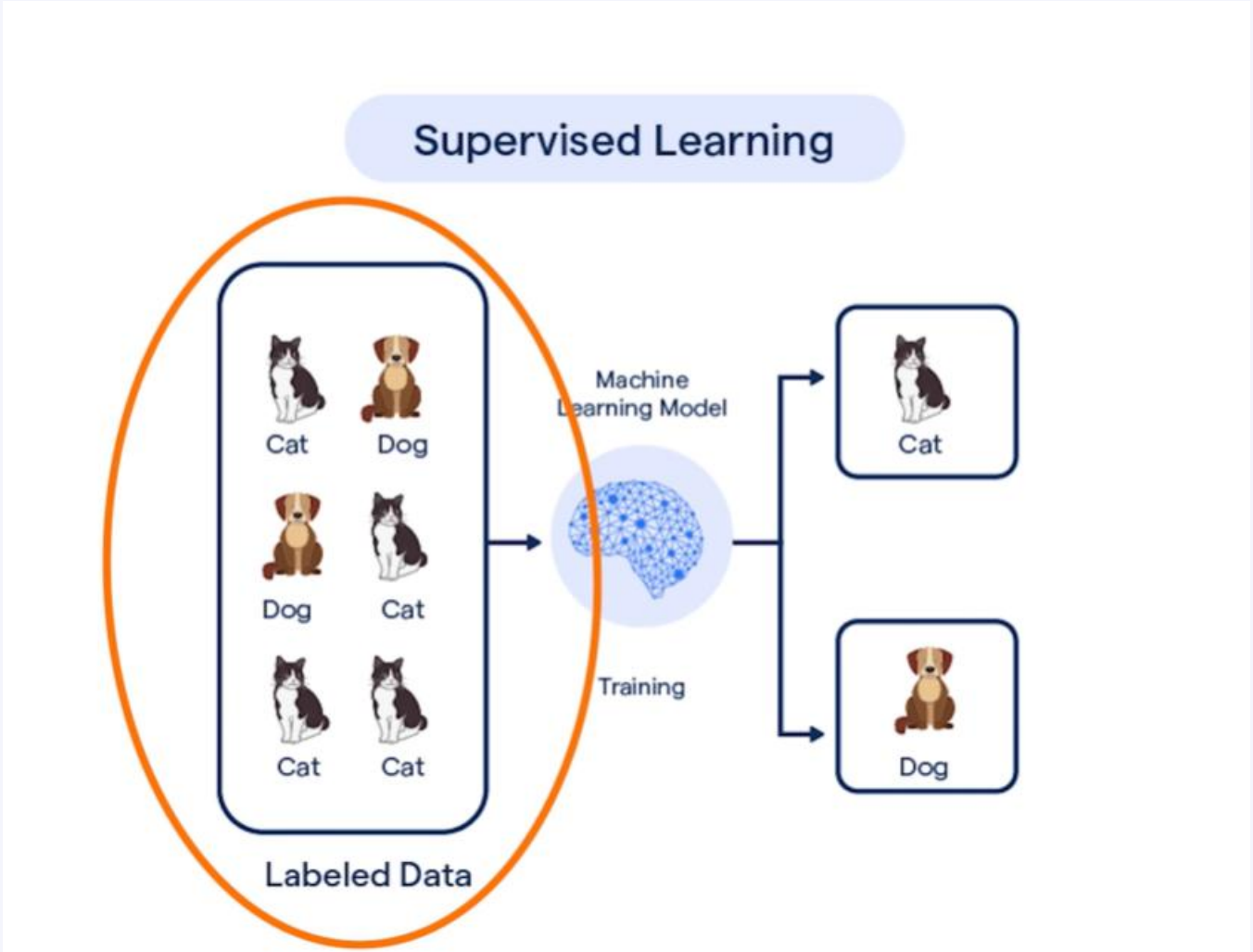

8. TEST DATA

Random split before modeling **begins**

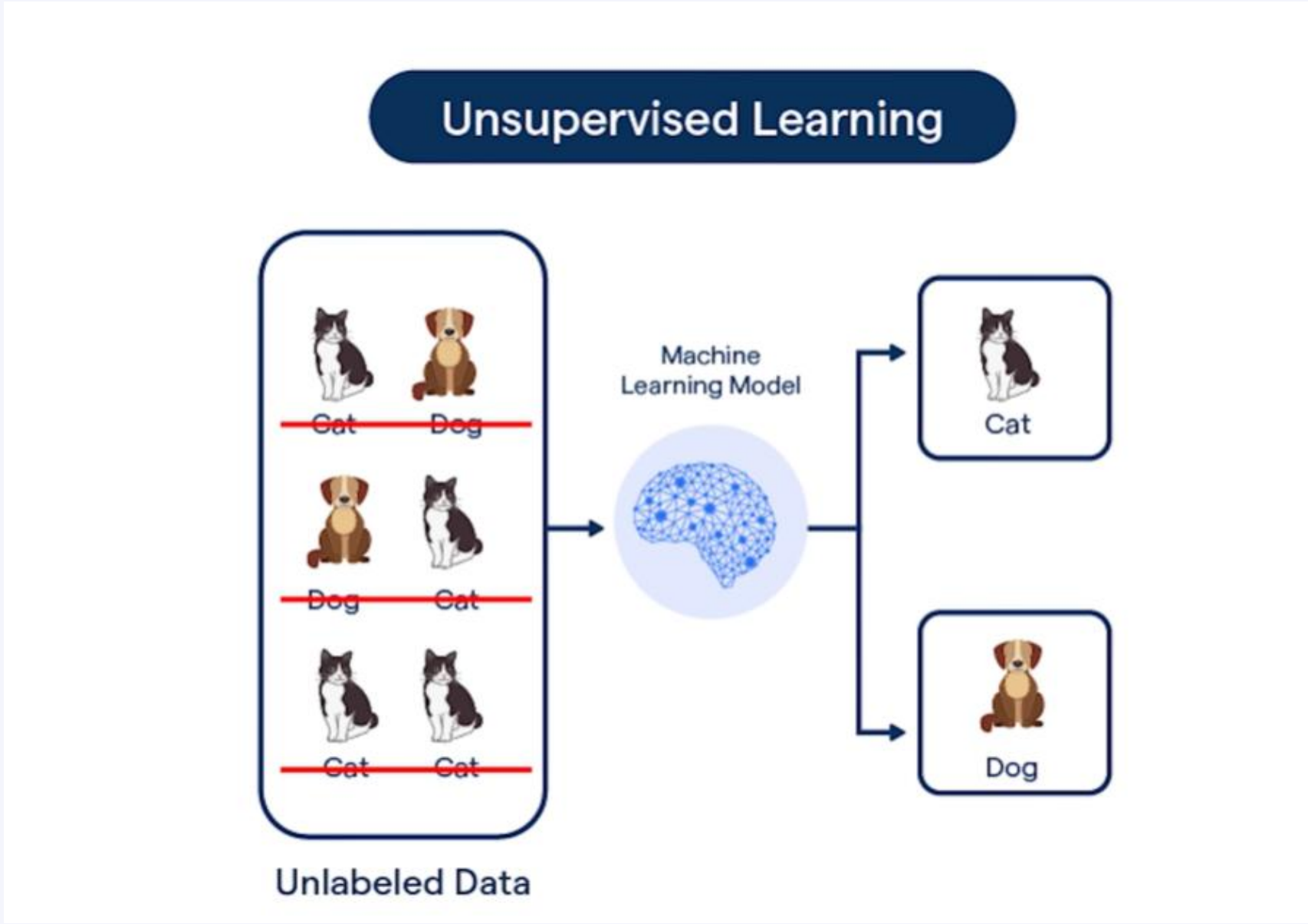
Random split before modeling **begins**

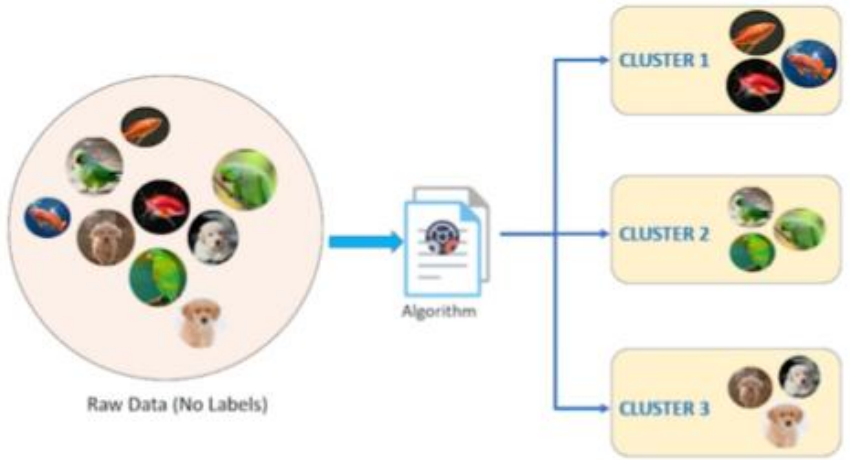

Data leakage
A
Training
Test
Single Dataset


9. SUPERVISED
LEARNING


Supervised Learning
Cat
Dog
Dog
Cat
Cat
Cat
Labeled Data
Machine
Learning Model
Training
Cat
Dog


10. UNSUPERVISED
LEARNING

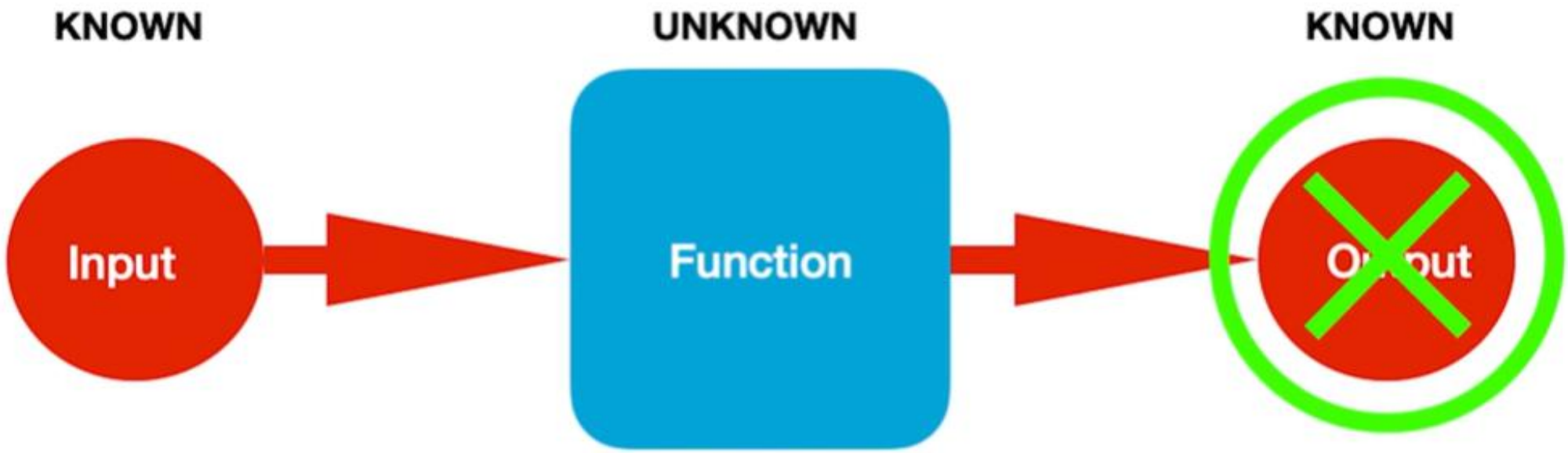

Unsupervised Learning
Cat
Dog
Dog
Cat
Cat
Cat
Unlabeled Data
Machine
Learning Model
Cat
Dog



Unsupervised Machine Learning

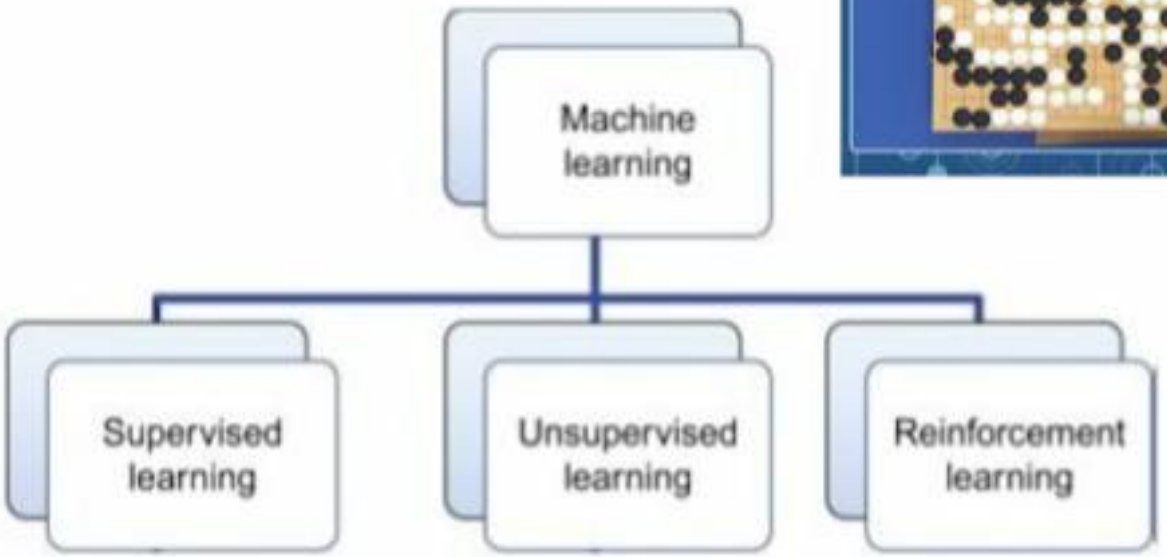
Unsupervised Learning examples

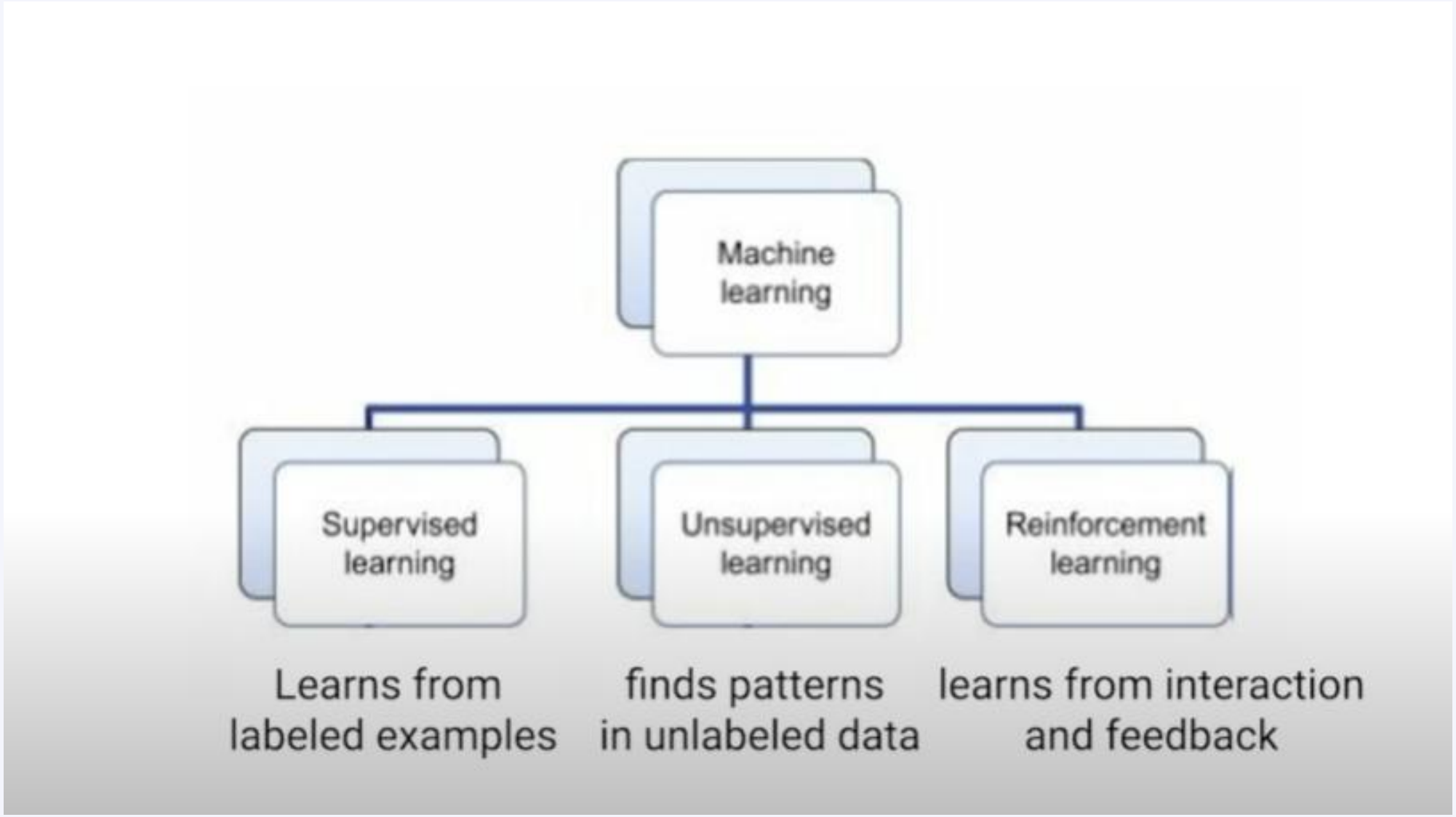
- Customer segregation
- Topic discovery
- When we don’t know what we’re looking for

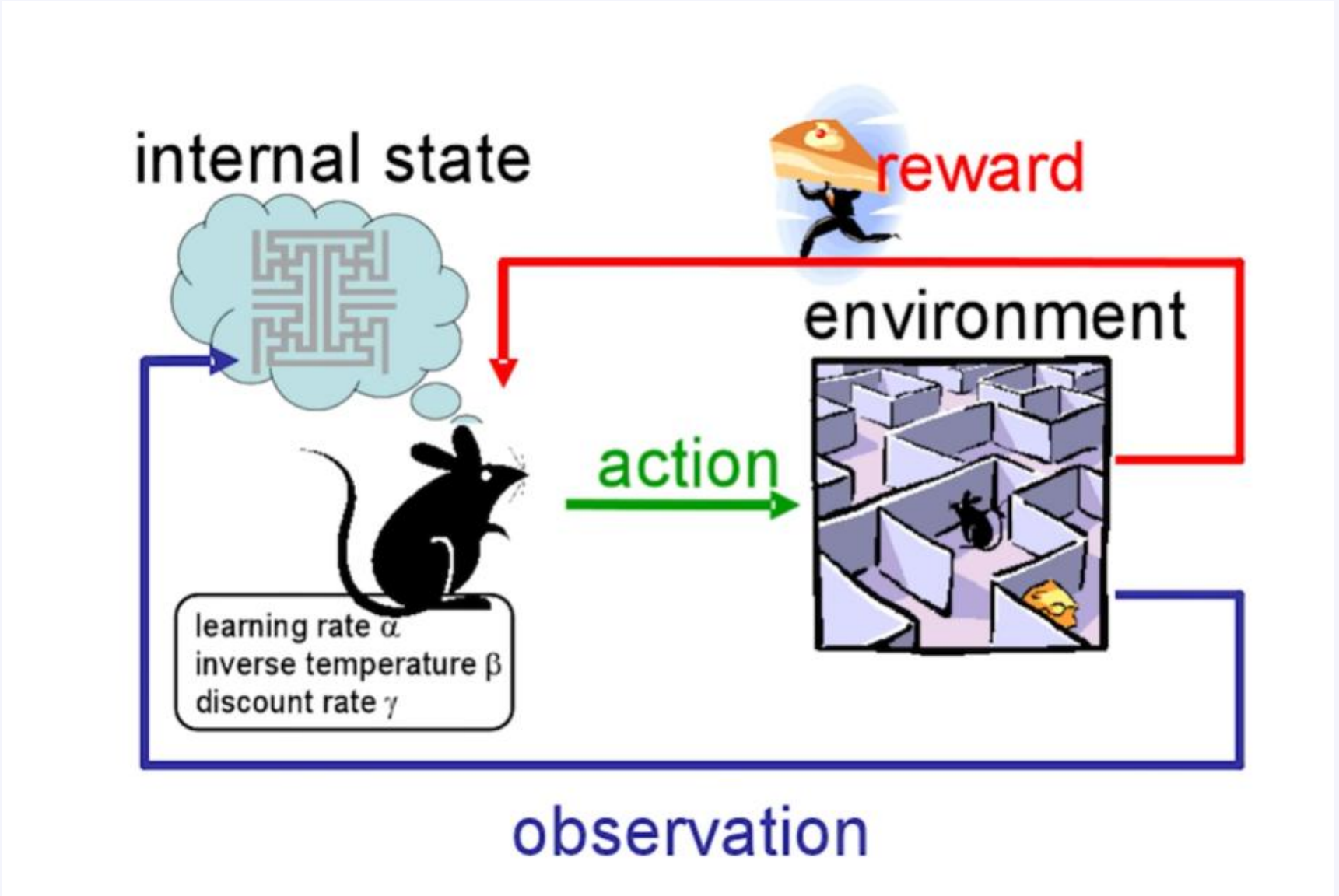

KNOWN
UNKNOWN
KNOWN
Input
Function

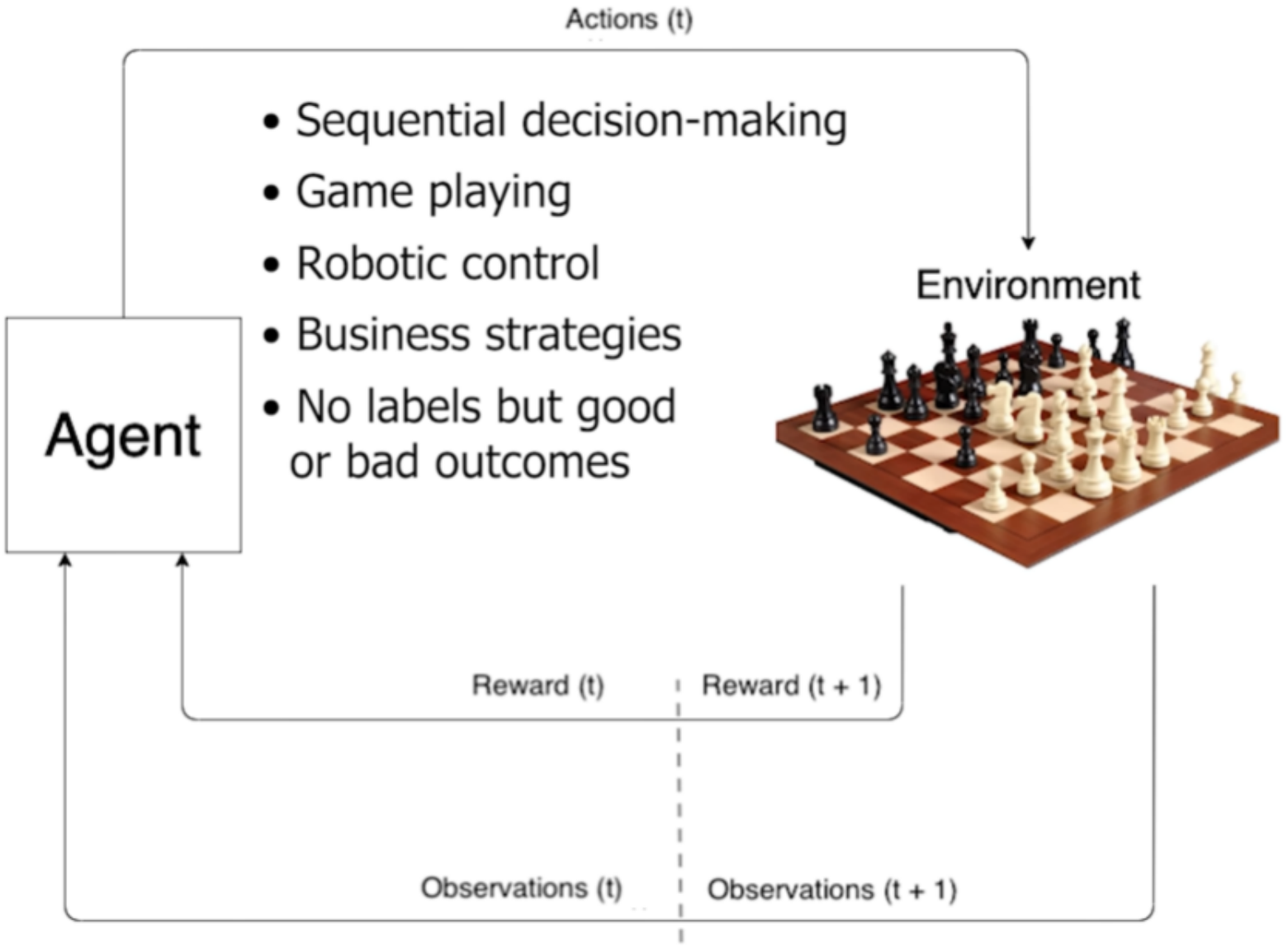

11. REINFORCEMENT
LEARNING


柯洁 KE JIE
00:13:28
ALPHAGO
01:30:35

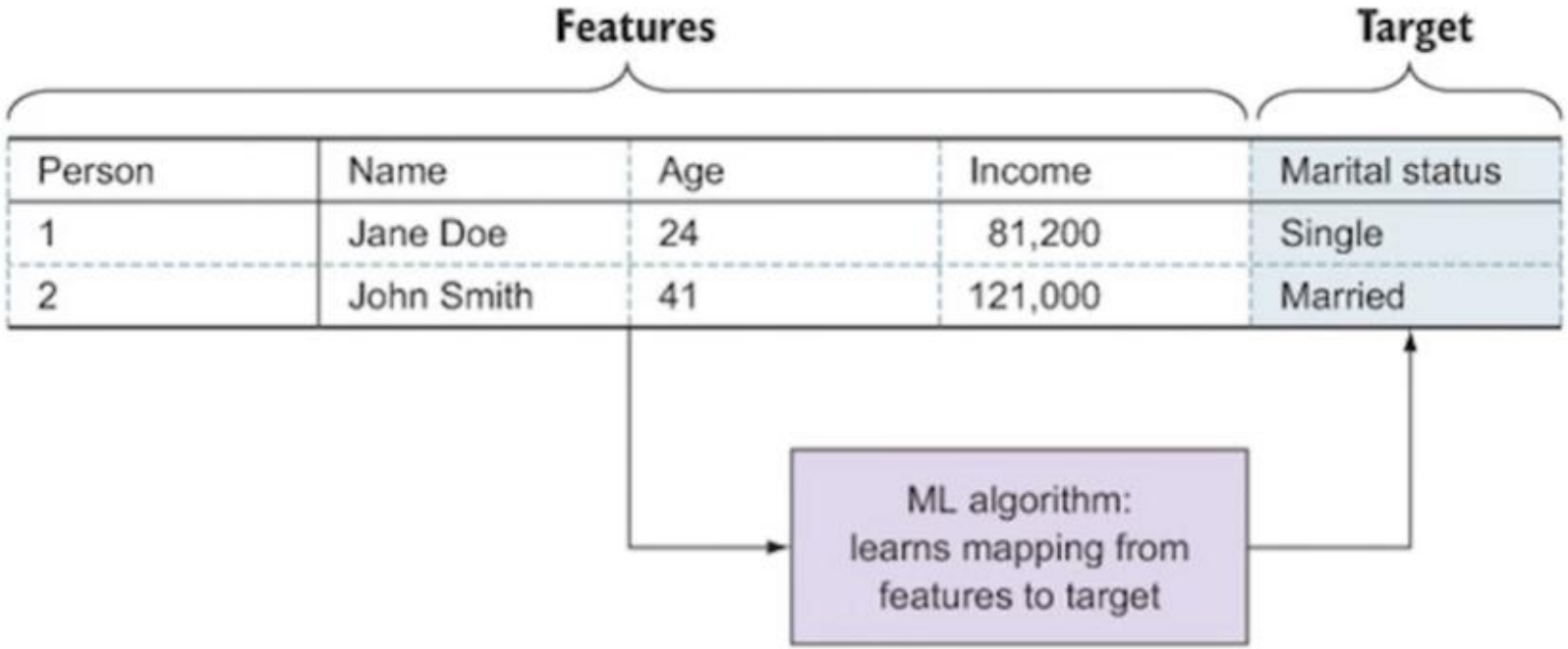

Machine learning
Supervised learning
Unsupervised learning
Reinforcement learning


Machine learning
Supervised learning
Unsupervised learning
Reinforcement learning
Learns from labeled examples
finds patterns in unlabeled data
learns from interaction and feedback


internal state
reward
environment
action
learning rate α
inverse temperature β
discount rate γ
observation


Actions (t)
• Sequential decision-making
• Game playing
• Robotic control
• Business strategies
• No labels but good or bad outcomes
Agent
Environment
Reward (t)
Reward (t + 1)
Observations (t)
Observations (t + 1)


12. FEATURE
(INPUT, INDEPENDENT
VARIABLE, PREDICTOR)



- Input
- Independent variable
- Attribute

features

	A	B	C	D	E
1	area	bedrooms	balcony	age	price
2	1200	2	0	2	500000
3	2300	3	2	5	620000
4	2500	4	2	1	122500
5	3650	5	3	3	6000000
6	1800	3	1	5	2122000
7	3000	3	1	4	120000
8	1222	1	0	2	450000
9	4600	5	3	1	6500000
10	2050	2	2	2	1530000
11	1450	2	2	3	1563330

Spam Attachments Features					
Habul Dataset			Botnet Dataset		
Rank	Category	Feature	Rank	Category	Feature
1	Subject	Number of capitalized words	1	Subject	Min of the compression ratio for the bz2 compressor
2	Subject	Sum of all the character lengths of words	2	Subject	Min of the compression ratio for the zlib compressor
3	Subject	Number of words containing letters and numbers	3	Subject	Min of character diversity of each word
4	Subject	Max of ratio of digit characters to all characters of each word	4	Subject	Min of the compression ratio for the lzw compressor
5	Header	Hour of day when email was sent	5	Subject	Max of the character lengths of words
		(a)			(b)
Spam URLs Features					
1	URL	The number of all URLs in an email	1	Header	Day of week when email was sent
2	URL	The number of unique URLs in an email	2	Payload	Number of characters
3	Payload	Number of words containing letters and numbers	3	Payload	Sum of all the character lengths of words
4	Payload	Min of the compression ratio for the bz2 compressor	4	Header	Minute of hour when email was sent
5	Payload	Number of words containing only letters	5	Header	Hour of day when email was sent
		(c)			(d)


13. FEATURE
ENGINEERING

Raw Data

Date
Jan 1, 2024
Feb 3, 2024
Mar 5, 2024

Engineered features

Day of the week	*Is Holiday?*
Sunday	Yes
Monday	No
Thursday	No


14. FEATURE SCALING
(NORMALIZATION,
STANDARDIZATION)

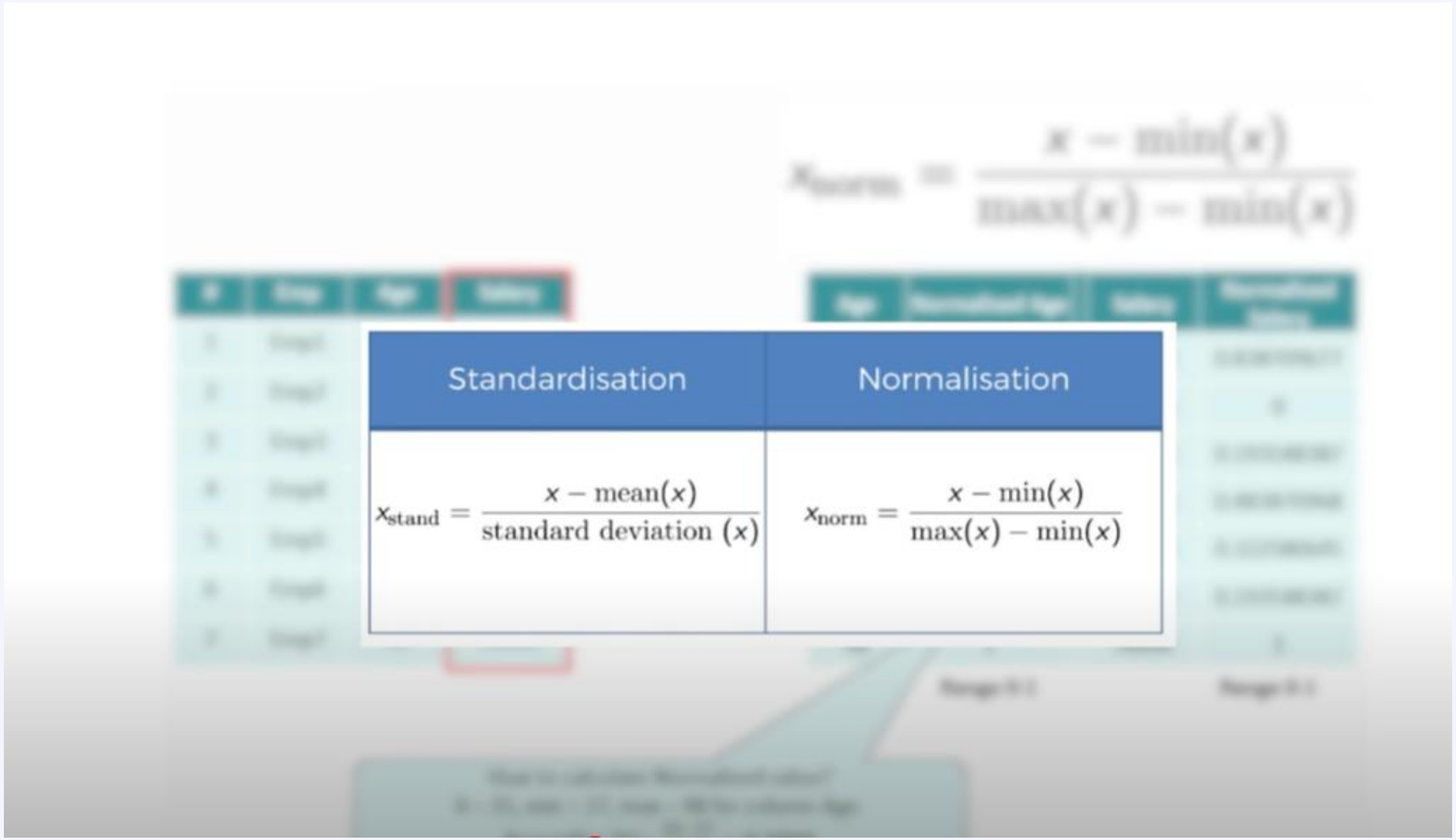
$$x_{\text{norm}} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

#	Emp	Age	Salary
1	Emp1	44	73000
2	Emp2	27	47000
3	Emp3	30	53000
4	Emp4	38	62000
5	Emp5	40	57000
6	Emp6	35	53000
7	Emp7	48	78000

Normalization

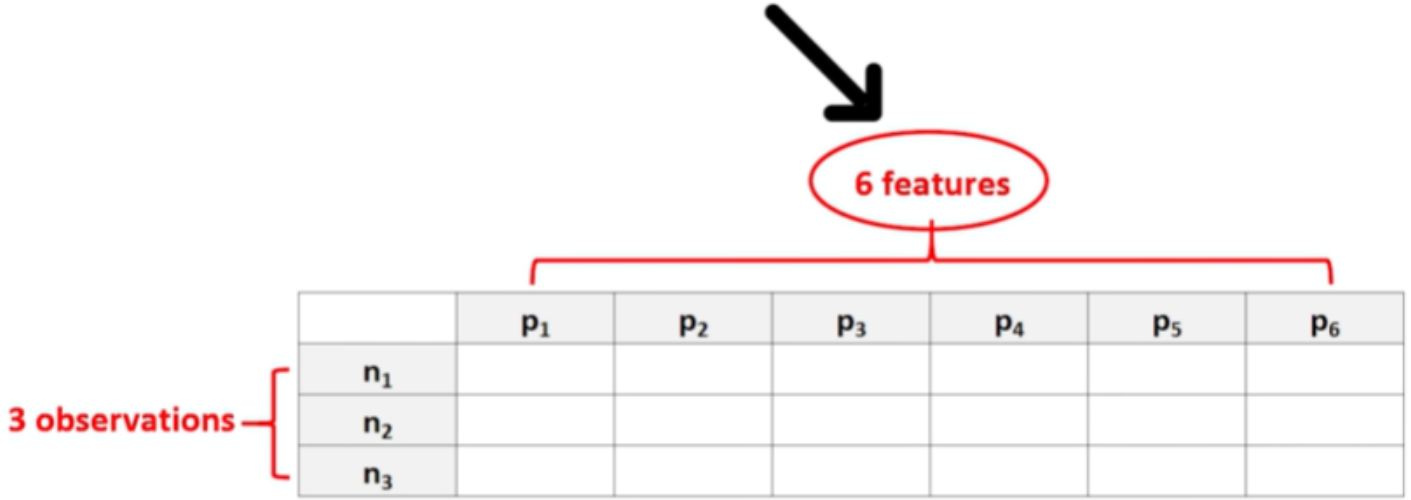
Age	Normalized Age	Salary	Normalized Salary
44	0.80952381	73000	0.838709677
27	0	47000	0
30	0.142857143	53000	0.193548387
38	0.523809524	62000	0.483870968
40	0.619047619	57000	0.322580645
35	0.380952381	53000	0.193548387
48	1	78000	1
	Range 0-1		Range 0-1

How to calculate Normalized value?
X = 35, min = 27, max = 48 for column Age.
Xnorm(for 35) = $\frac{35-27}{48-27} = 0.3809$


Standardisation
Normalisation
$x_{stand} = \frac{x - mean(x)}{standard\ deviation\ (x)}$
$x_{norm} = \frac{x - min(x)}{max(x) - min(x)}$


15. DIMENSIONALITY

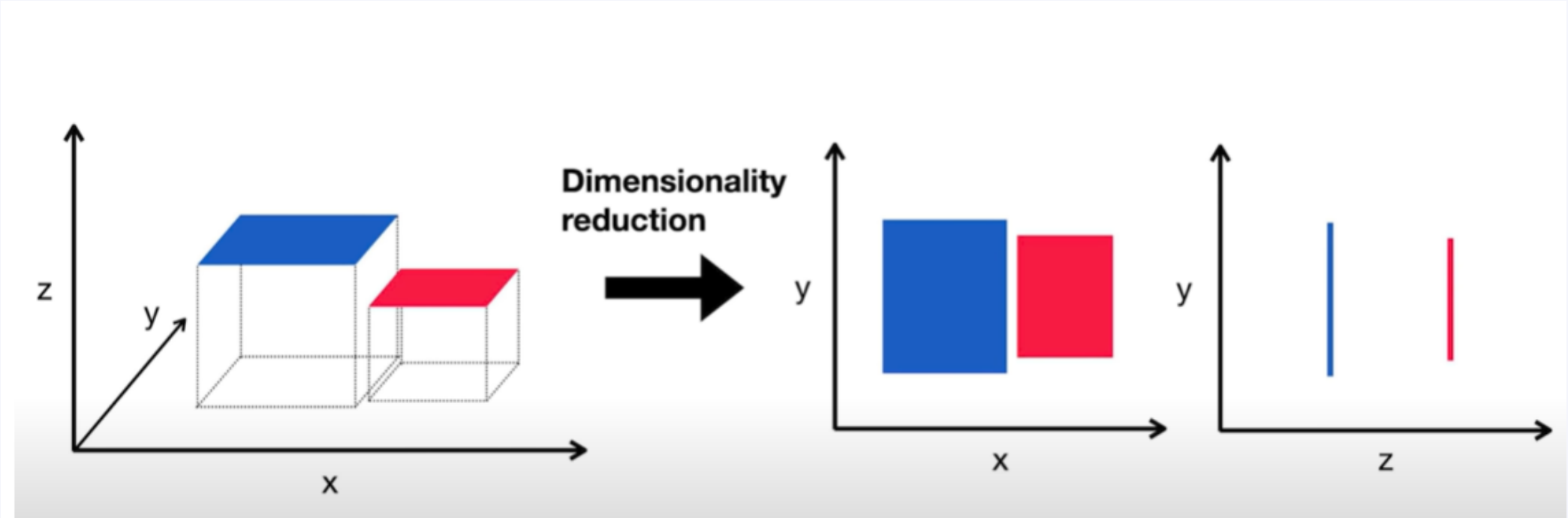
Dimensionality = 6



- SQFT
- No.of Bedroom
- Location
- Age
- No.of Bathrooms
- Distance

DrainI(1)	DrainV(1)	GateI(1)	GateV(1)	DrainI(2)	DrainV(2)	GateI(2)	GateV(2)	DrainI(3)	DrainV(3)	GateI(3)	GateV(3)	DrainI(4)	DrainV(4)	GateI(4)	GateV(4)	DrainI(5)	DrainV(5)	GateI(5)	GateV(5)
3.29E-07	0.00E+00	1.52E-07	1.00E+00	2.97E-07	0.00E+00	1.57E-07	9.00E-01	2.57E-07	0.00E+00	1.83E-07	8.00E-01	2.19E-07	0.00E+00	1.62E-07	7.00E-01	1.83E-07	0.00E+00	1.42E-07	6.00E-01
4.33E-06	1.00E-02	1.93E-07	1.00E+00	4.24E-06	1.00E-02	1.77E-07	9.00E-01	4.14E-06	1.00E-02	1.72E-07	8.00E-01	4.01E-06	1.00E-02	1.88E-07	7.00E-01	3.87E-06	1.00E-02	1.52E-07	6.00E-01
1.27E-05	2.00E-02	1.47E-07	1.00E+00	1.22E-05	2.00E-02	1.22E-07	9.00E-01	1.17E-05	2.00E-02	1.52E-07	8.00E-01	1.11E-05	2.00E-02	1.88E-07	7.00E-01	1.05E-05	2.00E-02	2.18E-07	6.00E-01
1.91E-05	3.00E-02	1.42E-07	1.00E+00	1.83E-05	3.00E-02	1.62E-07	9.00E-01	1.75E-05	3.00E-02	1.62E-07	8.00E-01	1.66E-05	3.00E-02	1.47E-07	7.00E-01	1.56E-05	3.00E-02	1.62E-07	6.00E-01
2.55E-05	4.00E-02	1.32E-07	1.00E+00	2.44E-05	4.00E-02	8.11E-08	9.00E-01	2.33E-05	4.00E-02	2.38E-07	8.00E-01	2.21E-05	4.00E-02	1.62E-07	7.00E-01	2.08E-05	4.00E-02	1.27E-07	6.00E-01
3.18E-05	5.00E-02	1.01E-07	1.00E+00	3.05E-05	5.00E-02	1.27E-07	9.00E-01	2.91E-05	5.00E-02	1.52E-07	8.00E-01	2.76E-05	5.00E-02	1.22E-07	7.00E-01	2.60E-05	5.00E-02	1.98E-07	6.00E-01
3.81E-05	6.00E-02	1.01E-07	1.00E+00	3.66E-05	6.00E-02	1.17E-07	9.00E-01	3.49E-05	6.00E-02	1.62E-07	8.00E-01	3.31E-05	6.00E-02	1.52E-07	7.00E-01	3.11E-05	6.00E-02	1.52E-07	6.00E-01
4.45E-05	7.00E-02	6.08E-08	1.00E+00	4.26E-05	7.00E-02	1.72E-07	9.00E-01	4.07E-05	7.00E-02	1.62E-07	8.00E-01	3.85E-05	7.00E-02	1.06E-07	7.00E-01	3.62E-05	7.00E-02	1.17E-07	6.00E-01
5.07E-05	8.00E-02	1.52E-07	1.00E+00	4.86E-05	8.00E-02	1.17E-07	9.00E-01	4.64E-05	8.00E-02	1.52E-07	8.00E-01	4.39E-05	8.00E-02	1.98E-07	7.00E-01	4.13E-05	8.00E-02	1.57E-07	6.00E-01
5.71E-05	9.00E-02	1.62E-07	1.00E+00	5.47E-05	9.00E-02	1.22E-07	9.00E-01	5.21E-05	9.00E-02	1.62E-07	8.00E-01	4.94E-05	9.00E-02	1.88E-07	7.00E-01	4.63E-05	9.00E-02	1.67E-07	6.00E-01
6.34E-05	1.00E-01	7.61E-08	1.00E+00	6.07E-05	1.00E-01	1.72E-07	9.00E-01	5.79E-05	1.00E-01	1.93E-07	8.00E-01	5.47E-05	1.00E-01	2.03E-07	7.00E-01	5.14E-05	1.00E-01	1.62E-07	6.00E-01
6.97E-05	1.10E-01	1.57E-07	1.00E+00	6.68E-05	1.10E-01	1.47E-07	9.00E-01	6.36E-05	1.10E-01	1.42E-07	8.00E-01	6.01E-05	1.10E-01	2.43E-07	7.00E-01	5.65E-05	1.10E-01	1.27E-07	6.00E-01
7.60E-05	1.20E-01	1.27E-07	1.00E+00	7.29E-05	1.20E-01	1.47E-07	9.00E-01	6.93E-05	1.20E-01	1.37E-07	8.00E-01	6.56E-05	1.20E-01	1.77E-07	7.00E-01	6.15E-05	1.20E-01	1.32E-07	6.00E-01
8.24E-05	1.30E-01	1.06E-07	1.00E+00	7.88E-05	1.30E-01	1.32E-07	9.00E-01	7.51E-05	1.30E-01	1.47E-07	8.00E-01	7.09E-05	1.30E-01	1.83E-07	7.00E-01	6.64E-05	1.30E-01	1.57E-07	6.00E-01
8.86E-05	1.40E-01	1.12E-07	1.00E+00	8.48E-05	1.40E-01	1.42E-07	9.00E-01	8.07E-05	1.40E-01	1.42E-07	8.00E-01	7.62E-05	1.40E-01	1.67E-07	7.00E-01	7.15E-05	1.40E-01	1.12E-07	6.00E-01
9.50E-05	1.50E-01	1.66E-07	1.00E+00	9.08E-05	1.50E-01	1.83E-07	9.00E-01	8.64E-05	1.50E-01	1.62E-07	8.00E-01	8.16E-05	1.50E-01	2.38E-07	7.00E-01	7.64E-05	1.50E-01	1.47E-07	6.00E-01
1.01E-04	1.60E-01	1.11E-07	1.00E+00	9.68E-05	1.60E-01	1.32E-07	9.00E-01	9.21E-05	1.60E-01	1.88E-07	8.00E-01	8.69E-05	1.60E-01	1.96E-07	7.00E-01	8.13E-05	1.60E-01	1.42E-07	6.00E-01
1.08E-04	1.70E-01	1.22E-07	1.00E+00	2.70E-04	1.70E-01	1.32E-07	9.00E-01	9.78E-05	1.70E-01	2.22E-07	8.00E-01	9.22E-05	1.70E-01	1.88E-07	7.00E-01	8.64E-05	1.70E-01	1.88E-07	6.00E-01
1.14E-04	1.80E-01	6.59E-08	1.00E+00	1.60E-01	1.80E-01	1.32E-07	9.00E-01	1.03E-04	1.80E-01	8.00E-01	8.00E-01	9.75E-05	1.80E-01	1.83E-07	7.00E-01	9.15E-05	1.80E-01	1.93E-07	6.00E-01
1.20E-04	1.90E-01	1.22E-07	1.00E+00	1.15E-04	1.90E-01	1.22E-07	9.00E-01	1.09E-04	1.90E-01	1.57E-07	8.00E-01	1.03E-04	1.90E-01	2.33E-07	7.00E-01	9.61E-05	1.90E-01	1.57E-07	6.00E-01
1.27E-04	2.00E-01	1.32E-07	1.00E+00	1.21E-04	2.00E-01	1.32E-07	9.00E-01	1.15E-04	2.00E-01	1.12E-07	8.00E-01	1.08E-04	2.00E-01	2.43E-07	7.00E-01	1.01E-04	2.00E-01	1.47E-07	6.00E-01
1.33E-04	2.10E-01	1.77E-07	1.00E+00	1.27E-04	2.10E-01	1.32E-07	9.00E-01	1.20E-04	2.10E-01	1.32E-07	8.00E-01	1.13E-04	2.10E-01	1.57E-07	7.00E-01	1.06E-04	2.10E-01	1.83E-07	6.00E-01
1.39E-04	2.20E-01	1.42E-07	1.00E+00	1.33E-04	2.20E-01	1.37E-07	9.00E-01	1.26E-04	2.20E-01	1.88E-07	8.00E-01	1.19E-04	2.20E-01	1.88E-07	7.00E-01	1.11E-04	2.20E-01	1.52E-07	6.00E-01
1.45E-04	2.30E-01	1.32E-07	1.00E+00	1.39E-04	2.30E-01	1.88E-07	9.00E-01	1.32E-04	2.30E-01	1.62E-07	8.00E-01	1.24E-04	2.30E-01	1.52E-07	7.00E-01	1.16E-04	2.30E-01	1.57E-07	6.00E-01
1.52E-04	2.40E-01	1.22E-07	1.00E+00	1.45E-04	2.40E-01	1.88E-07	9.00E-01	1.37E-04	2.40E-01	1.77E-07	8.00E-01	1.29E-04	2.40E-01	1.32E-07	7.00E-01	1.20E-04	2.40E-01	1.42E-07	6.00E-01
1.58E-04	2.50E-01	8.62E-08	1.00E+00	1.51E-04	2.50E-01	2.03E-07	9.00E-01	1.43E-04	2.50E-01	1.47E-07	8.00E-01	1.34E-04	2.50E-01	1.62E-07	7.00E-01	1.25E-04	2.50E-01	1.27E-07	6.00E-01
1.64E-04	2.60E-01	1.12E-07	1.00E+00	1.57E-04	2.60E-01	1.77E-07	9.00E-01	1.49E-04	2.60E-01	1.47E-07	8.00E-01	1.39E-04	2.60E-01	1.72E-07	7.00E-01	1.30E-04	2.60E-01	1.12E-07	6.00E-01
1.71E-04	2.70E-01	9.13E-08	1.00E+00	1.63E-04	2.70E-01	1.52E-07	9.00E-01	1.54E-04	2.70E-01	1.57E-07	8.00E-01	1.45E-04	2.70E-01	1.12E-07	7.00E-01	1.35E-04	2.70E-01	1.67E-07	6.00E-01
1.77E-04	2.80E-01	1.01E-07	1.00E+00	1.69E-04	2.80E-01	1.42E-07	9.00E-01	1.60E-04	2.80E-01	1.52E-07	8.00E-01	1.50E-04	2.80E-01	1.06E-07	7.00E-01	1.39E-04	2.80E-01	1.52E-07	6.00E-01
1.83E-04	2.90E-01	1.17E-07	1.00E+00	1.75E-04	2.90E-01	1.12E-07	9.00E-01	1.65E-04	2.90E-01	1.37E-07	8.00E-01	1.55E-04	2.90E-01	9.63E-08	7.00E-01	1.44E-04	2.90E-01	1.77E-07	6.00E-01
1.90E-04	3.00E-01	6.59E-08	1.00E+00	1.81E-04	3.00E-01	1.57E-07	9.00E-01	1.71E-04	3.00E-01	1.72E-07	8.00E-01	1.60E-04	3.00E-01	1.17E-07	7.00E-01	1.49E-04	3.00E-01	1.57E-07	6.00E-01
1.96E-04	3.10E-01	6.08E-08	1.00E+00	1.87E-04	3.10E-01	1.32E-07	9.00E-01	1.76E-04	3.10E-01	1.27E-07	8.00E-01	1.65E-04	3.10E-01	1.06E-07	7.00E-01	1.54E-04	3.10E-01	1.22E-07	6.00E-01

CURSE OF DIMENSIONALITY

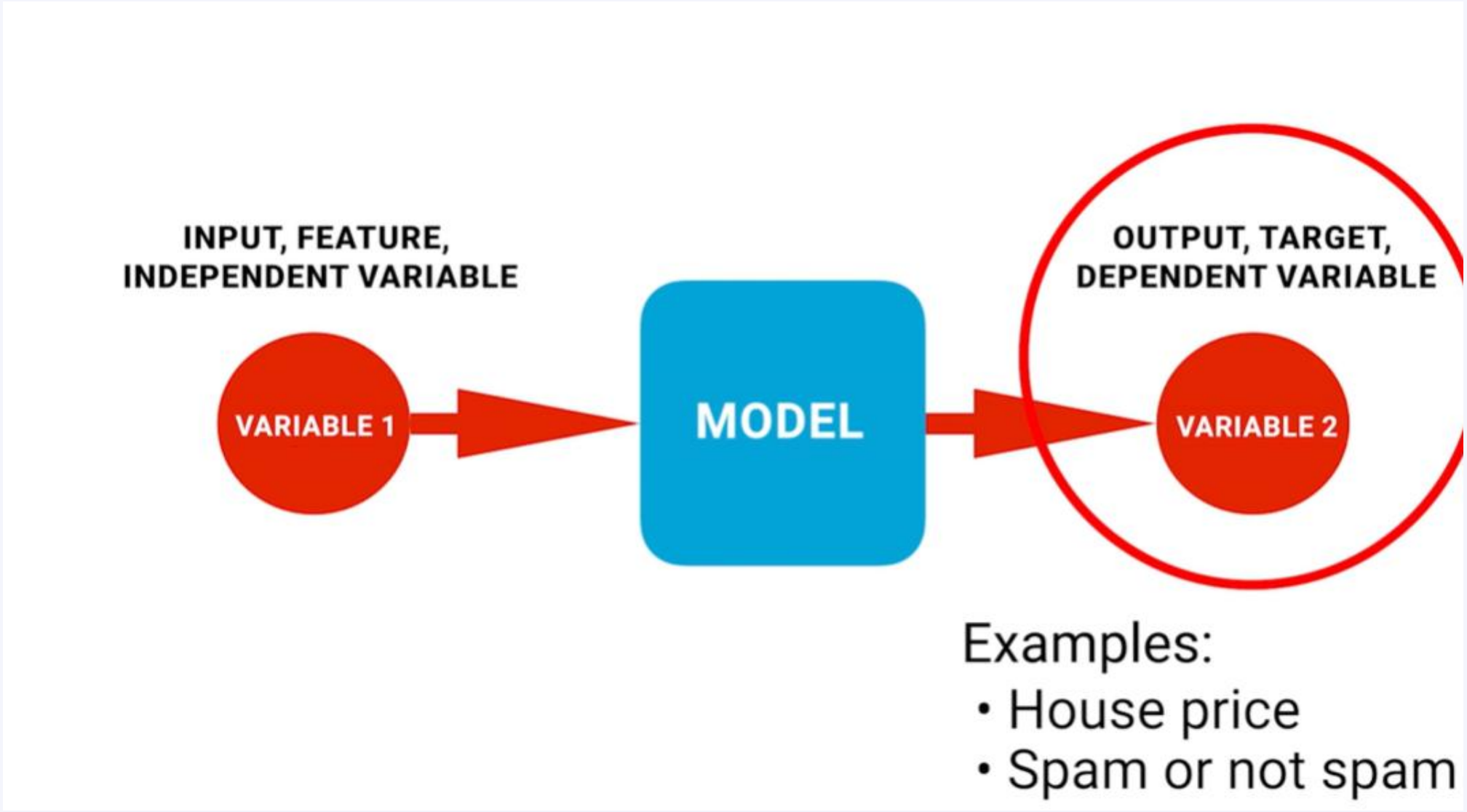

z
y
x
Dimensionality reduction
y
x
y
z

- Data preprocessing techniques:

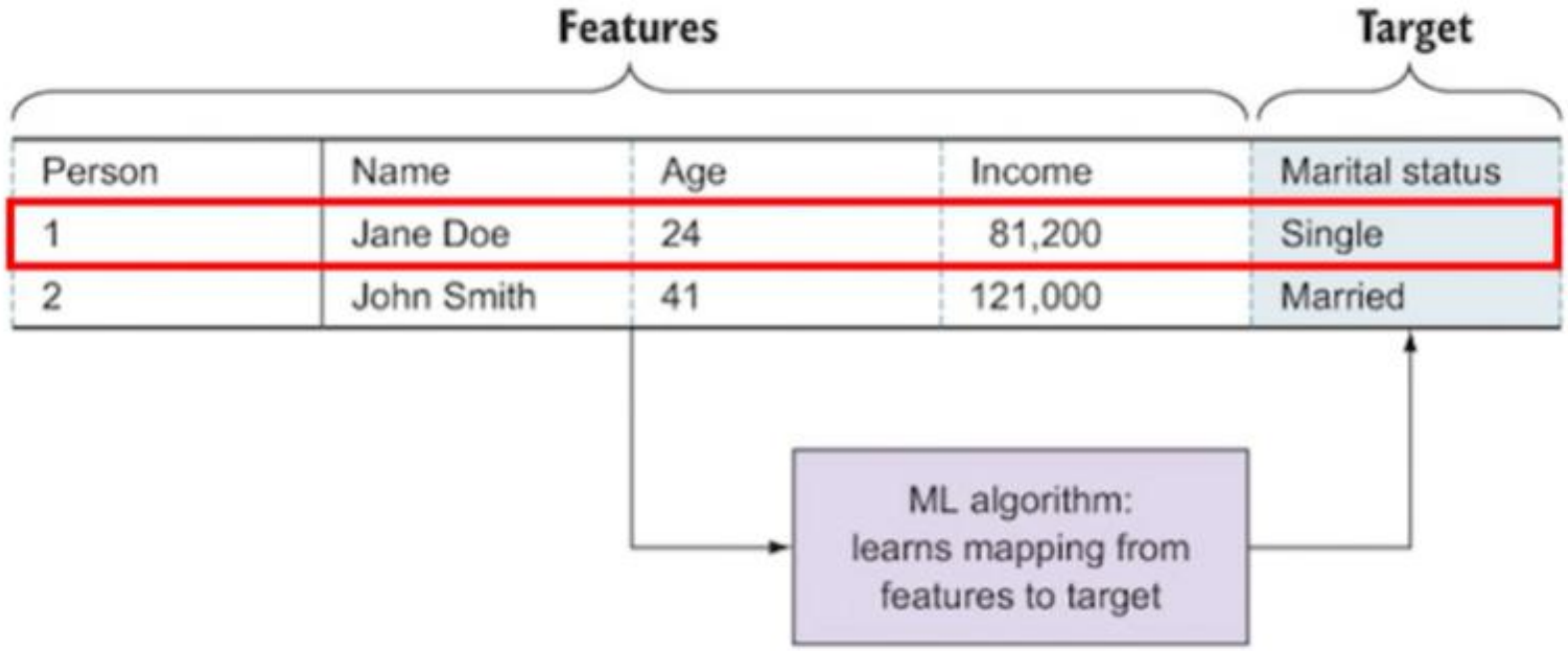
  - Feature Engineering
  - Feature Scaling
  - Dimensionality Reduction
  - Data Cleaning
  - Data Augmentation

  ...


16. TARGET
(OUTPUT, LABEL,
DEPENDENT VARIABLE)


INPUT, FEATURE,
INDEPENDENT VARIABLE
VARIABLE 1
MODEL
OUTPUT, TARGET,
DEPENDENT VARIABLE
VARIABLE 2
Examples:
• House price
• Spam or not spam


17. INSTANCE
(EXAMPLE, OBSERVATION,
SAMPLE)


Features
Target
Person
Name
Age
Income
Marital status
1
Jane Doe
24
81,200
Single
2
John Smith
41
121,000
Married
ML algorithm:
learns mapping from
features to target

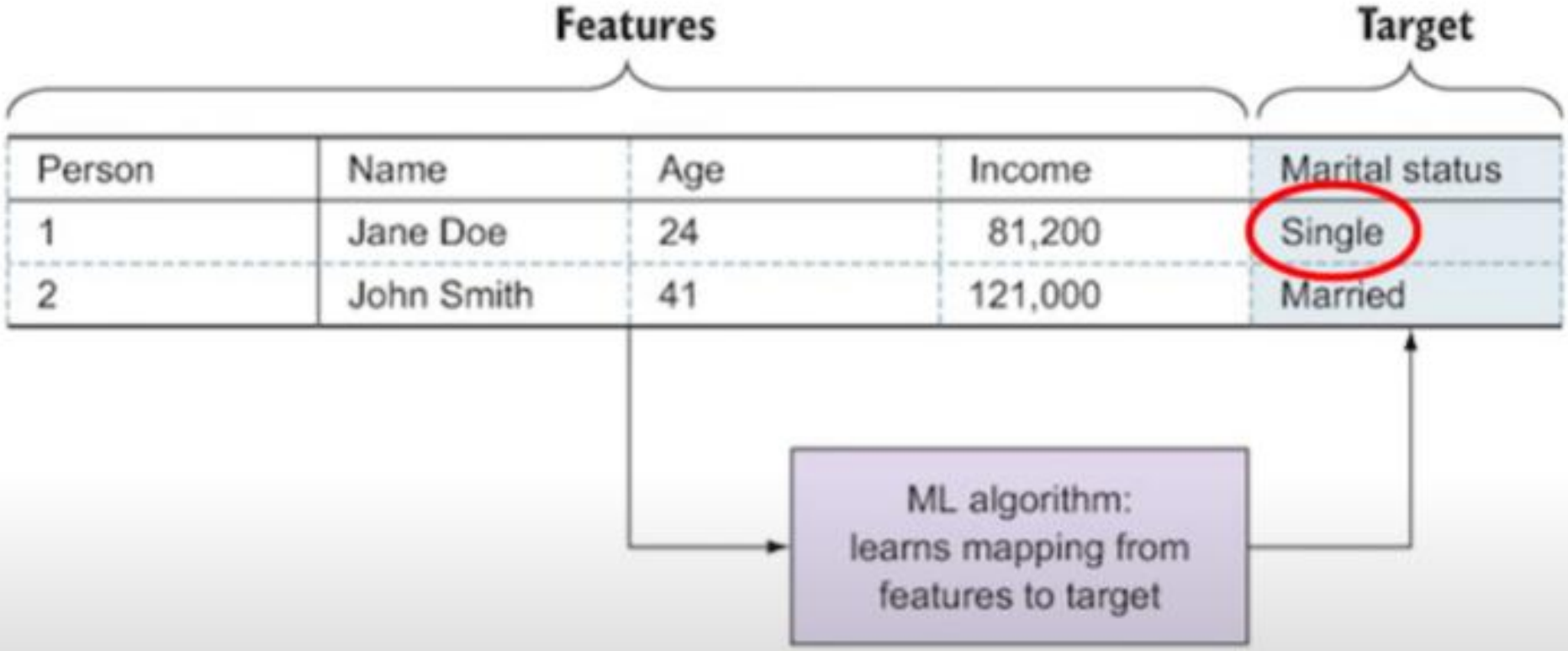
DATASET

TARGET

39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K
37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wife	White	Female	0	0	40	United-States	<=50K
49	Private	160187	9th	5	Married-spouse-absent	Other-service	Not-in-family	Black	Female	0	0	16	Jamaica	<=50K
52	Self-emp-not-inc	209642	HS-grad	9	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	45	United-States	>50K
31	Private	45781	Masters	14	Never-married	Prof-specialty	Not-in-family	White	Female	14084	0	50	United-States	>50K
42	Private	159449	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	5178	0	40	United-States	>50K
37	Private	280464	Some-college	10	Married-civ-spouse	Exec-managerial	Husband	Black	Male	0	0	80	United-States	>50K
30	State-gov	141297	Bachelors	13	Married-civ-spouse	Prof-specialty	Husband	Asian-Pac-Islander	Male	0	0	40	India	>50K
23	Private	122272	Bachelors	13	Never-married	Adm-clerical	Own-child	White	Female	0	0	30	United-States	<=50K
32	Private	205019	Assoc-acdm	12	Never-married	Sales	Not-in-family	Black	Male	0	0	50	United-States	<=50K
40	Private	121772	Assoc-voc	11	Married-civ-spouse	Craft-repair	Husband	Asian-Pac-Islander	Male	0	0	40	?	>50K
34	Private	245487	7th-8th	4	Married-civ-spouse	Transport-moving	Husband	Amer-Indian-Eskimo	Male	0	0	45	Mexico	<=50K
25	Self-emp-not-inc	176756	HS-grad	9	Never-married	Farming-fishing	Own-child	White	Male	0	0	35	United-States	<=50K
32	Private	186824	HS-grad	9	Never-married	Machine-op-inspct	Unmarried	White	Male	0	0	40	United-States	<=50K
38	Private	28887	11th	7	Married-civ-spouse	Sales	Husband	White	Male	0	0	50	United-States	<=50K
43	Self-emp-not-inc	292175	Masters	14	Divorced	Exec-managerial	Unmarried	White	Female	0	0	45	United-States	>50K
40	Private	193524	Doctorate	16	Married-civ-spouse	Prof-specialty	Husband	White	Male	0	0	60	United-States	>50K
54	Private	302146	HS-grad	9	Separated	Other-service	Unmarried	Black	Female	0	0	20	United-States	<=50K
35	Federal-gov	76845	9th	5	Married-civ-spouse	Farming-fishing	Husband	Black	Male	0	0	40	United-States	<=50K
43	Private	117037	11th	7	Married-civ-spouse	Transport-moving	Husband	White	Male	0	2042	40	United-States	<=50K
59	Private	109015	HS-grad	9	Divorced	Tech-support	Unmarried	White	Female	0	0	40	United-States	<=50K

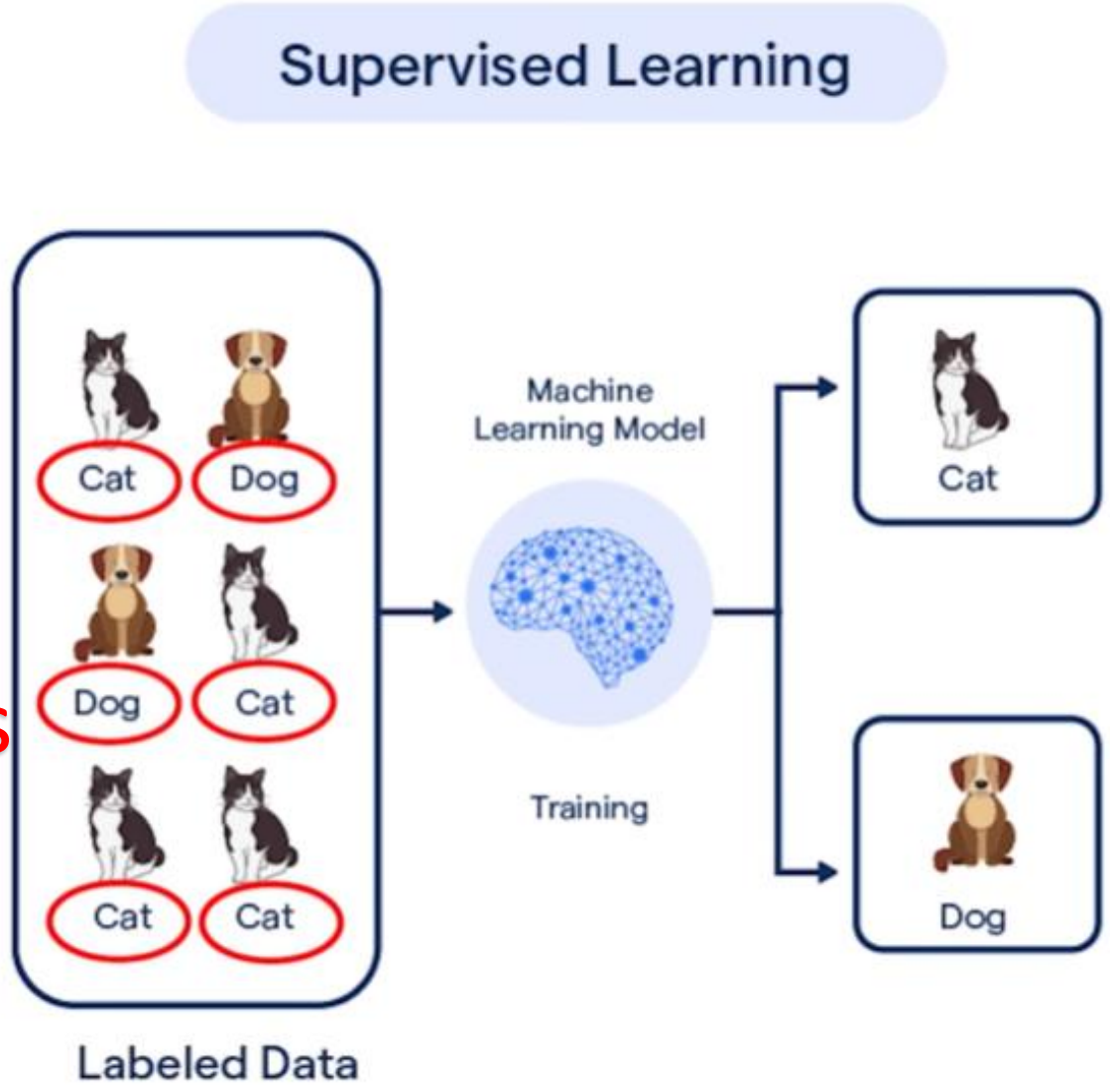
FEATURES

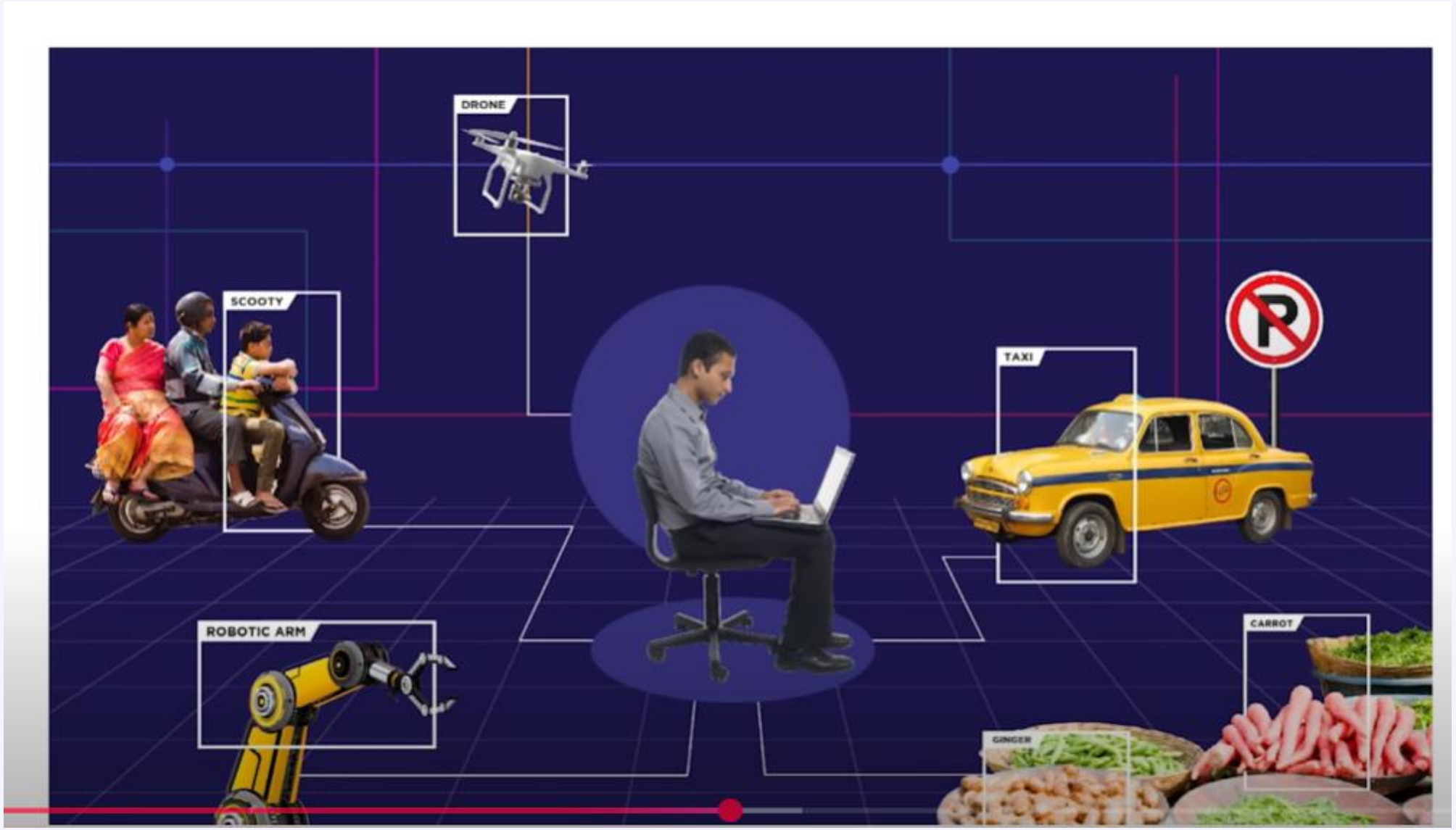

18. LABEL (CLASS,
TARGET VALUE)

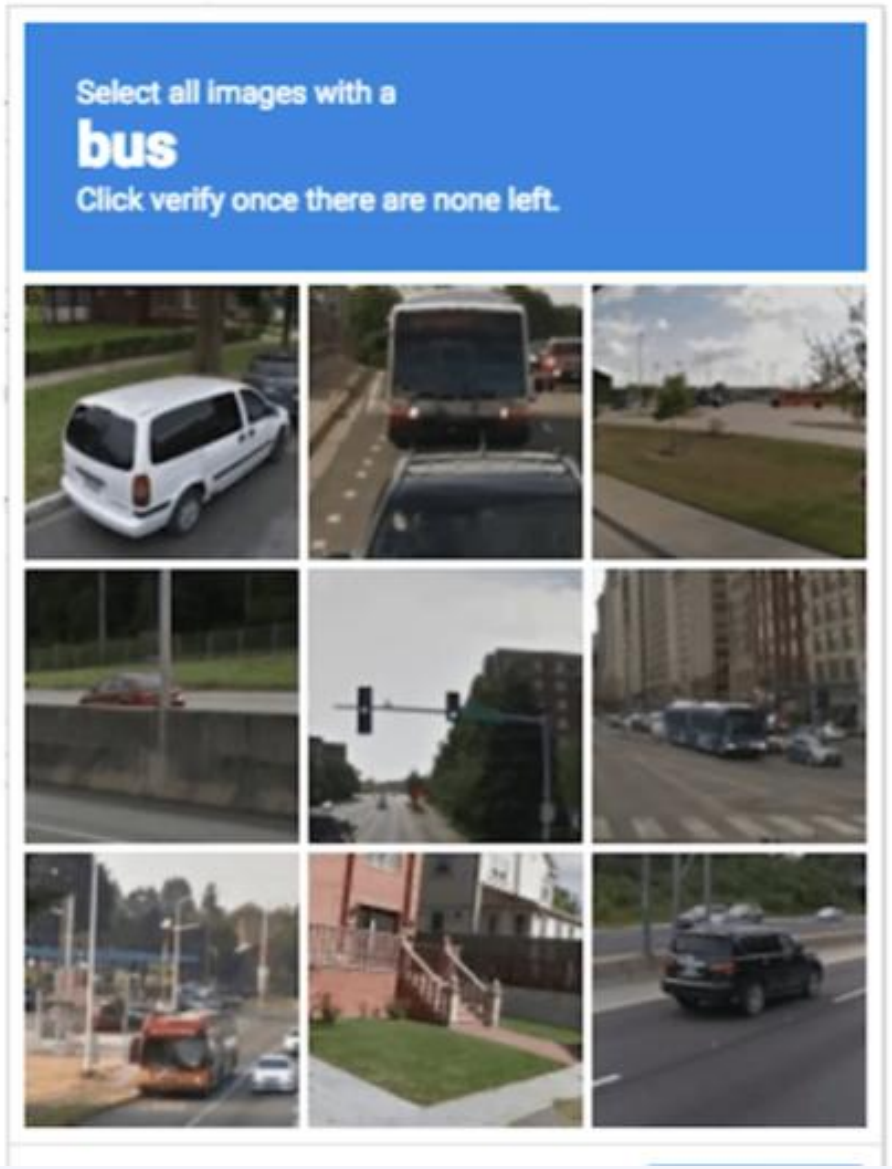

Features
Target
Person
Name
Age
Income
Marital status
1
Jane Doe
24
81,200
Single
2
John Smith
41
121,000
Married
ML algorithm:
learns mapping from
features to target

Labels

Right answers




DRONE
SCOOTY
TAXI
ROBOTIC ARM
CARROT
GINGER

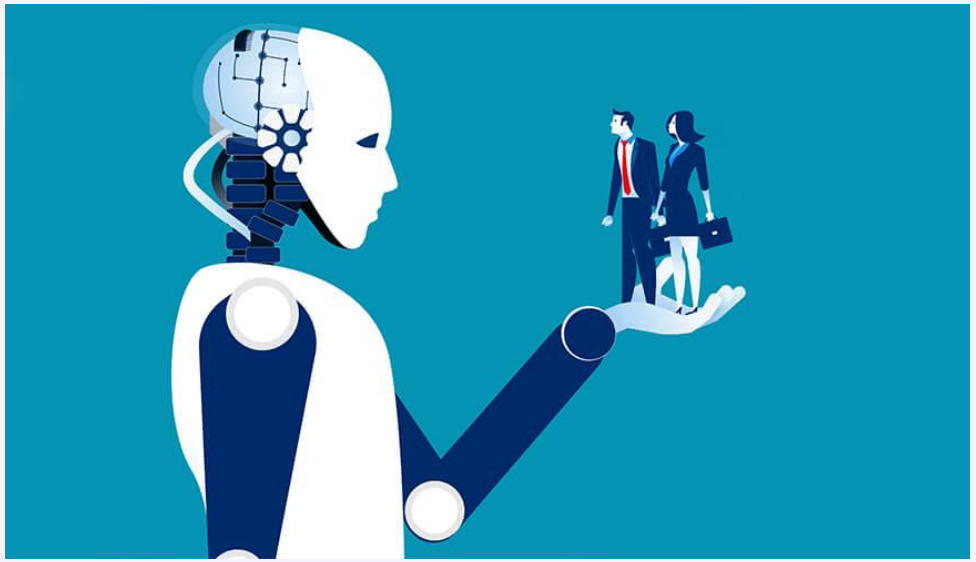


YES YOU ARE LABELING IMAGES FOR GOOGLE FOR FREE!

Let’s Get Started

Introduction

- Machine learning is making great strides
  - Large, good data sets
  - Compute power
  - Progress in algorithms
- Many interesting applications
  - commericial
  - scientific
- Links with artificial intelligence
  - However, AI ≠ machine learning



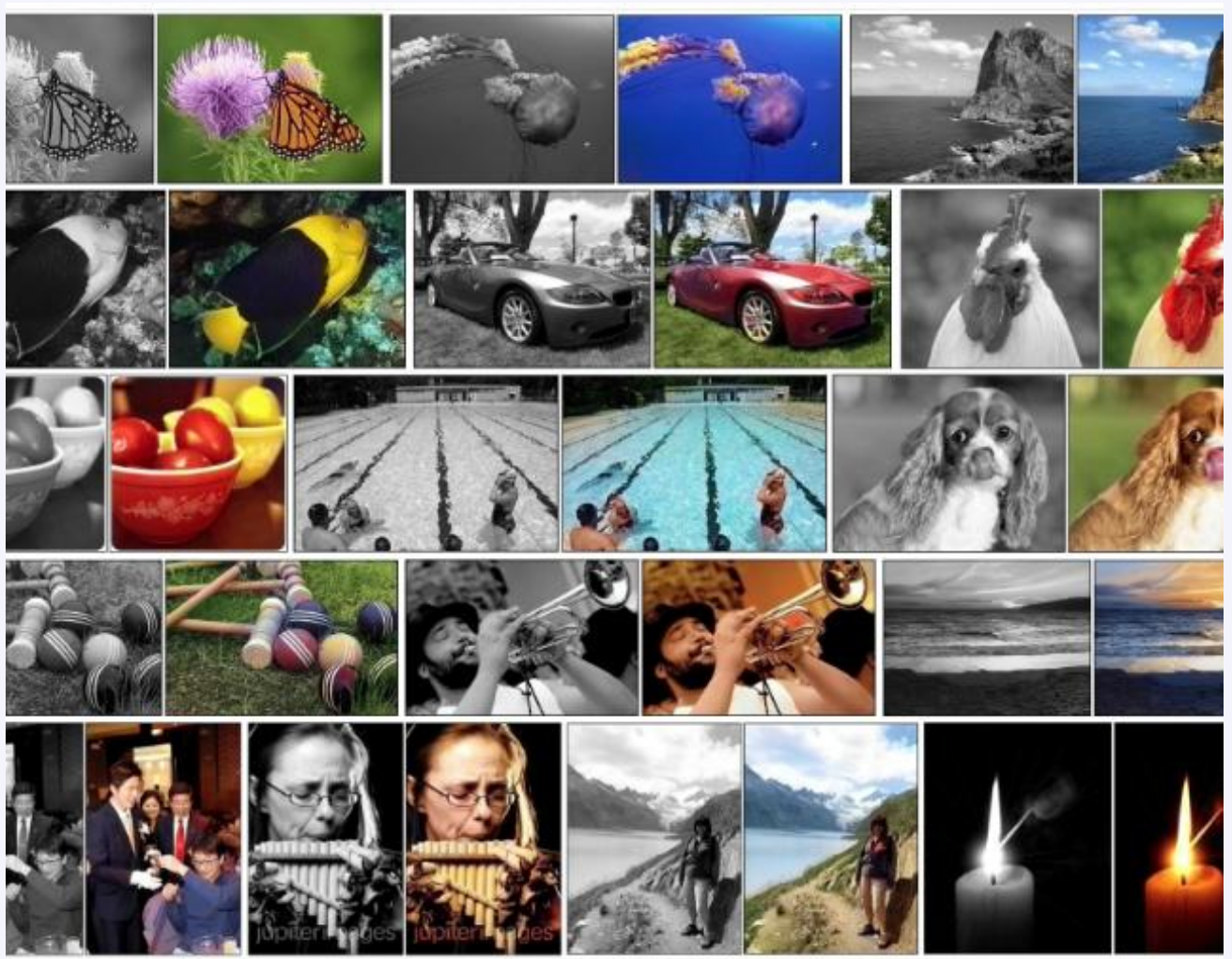
Machine learning tasks

- Supervised learning
  - regression: predict numerical values
  - classification: predict categorical values, i.e., labels
- Unsupervised learning
  - clustering: group data according to "distance"
  - association: find frequent co-occurrences
  - link prediction: discover relationships in data
  - data reduction: project features to fewer features
- Reinforcement learning

Regression

Colorize B&W images automatically

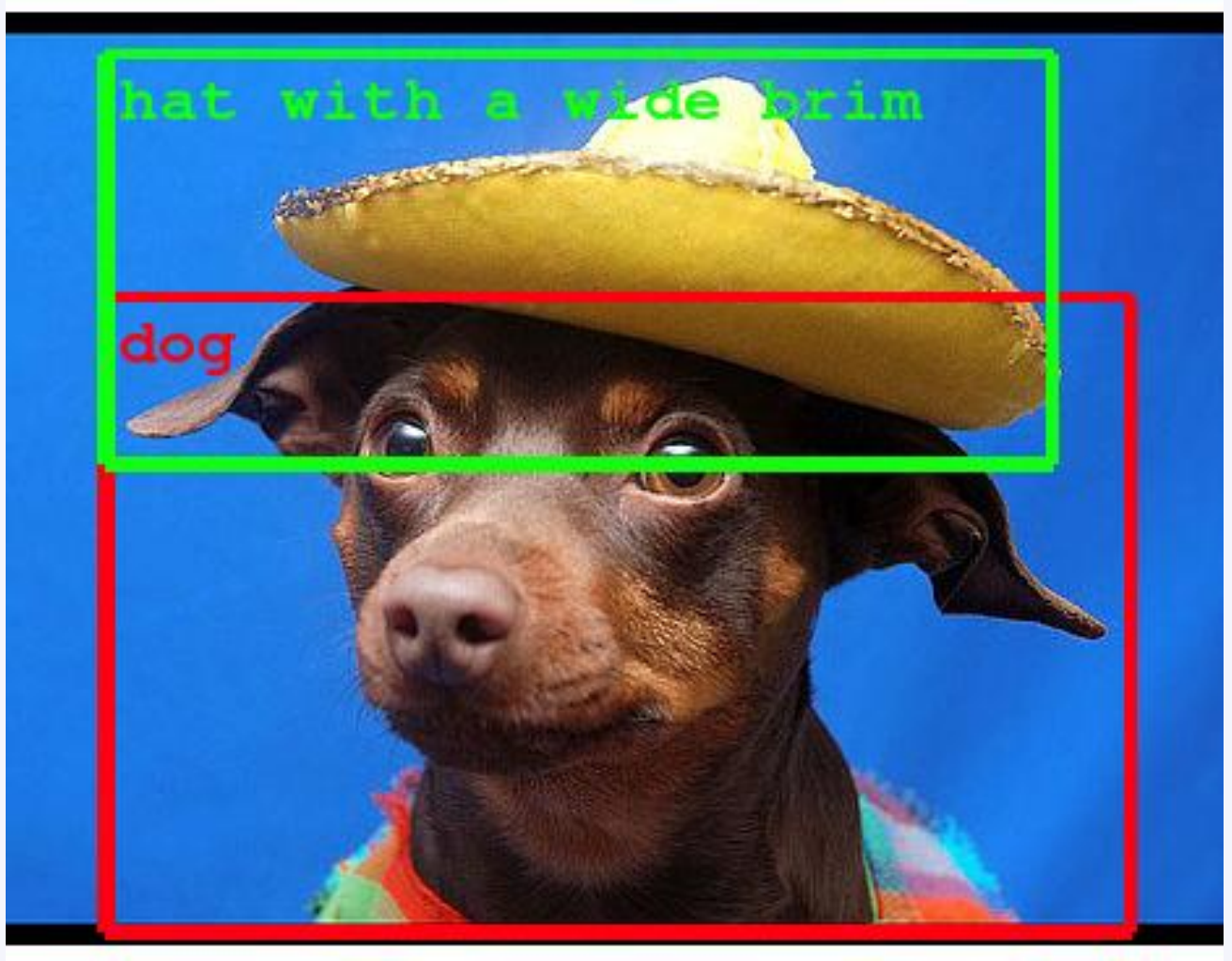
https://tinyclouds.org/colorize/



Classification

Object recognition

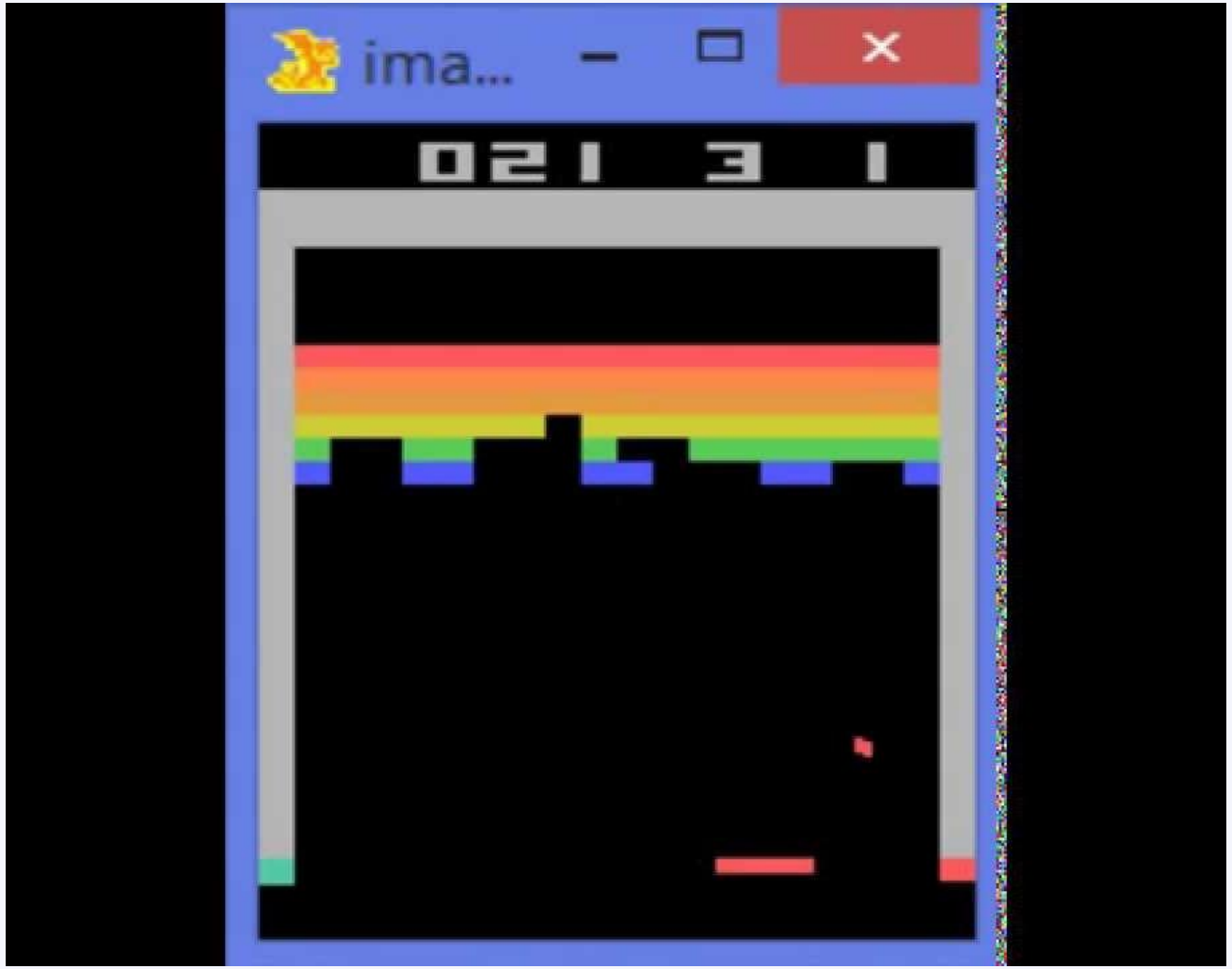
https://ai.googleblog.com/2014/09/building-deeper-understanding-of-images.html



Reinforcement learning

Learning to play Break Out

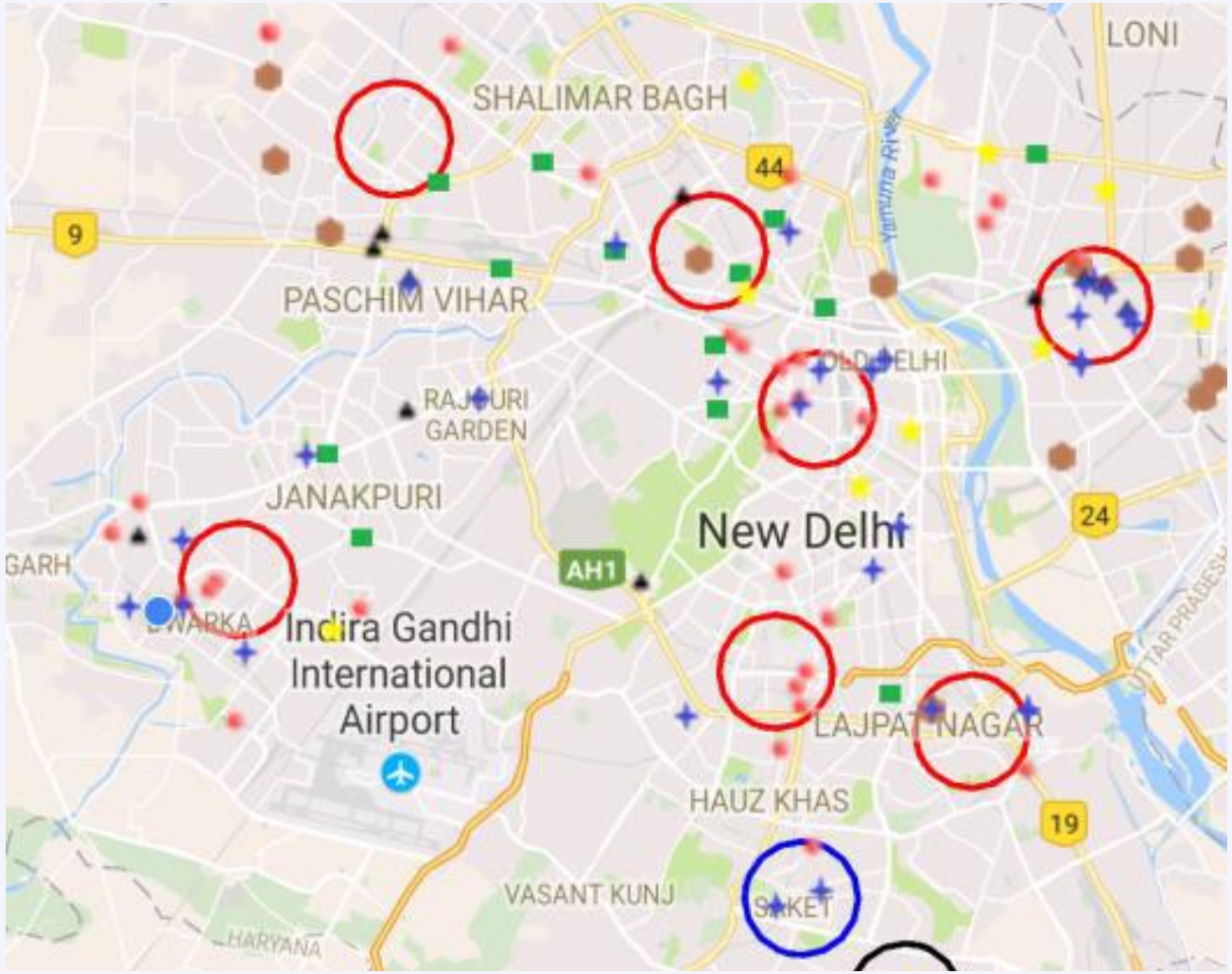
https://www.youtube.com/watch?v=V1eYniJ0Rnk



Clustering

Crime prediction using k-means clustering

http://www.grdjournals.com/uploads/article/GRDJE/V02/I05/0176/GRDJEV02I050176.pdf



Applications in science

Earth Observation Open Science and Innovation pp 165-218 | Cite as

Machine Learning Applications for Earth Observation

Authors and affiliations

Authors

David J. Lary, Gebreab K. Zewdie, Xun Liu, Daji Wu, Estelle Levetin, Rebecca J. Allee, Nabin Malakar, Annette Walker, Hamse Mussa, Antonio Mannino, Dirk Aurin

Chapter

4 Readers 17 Downloads 9.5k

nature
International journal of science

Review Article | Published: 25 July 2018

Machine learning for molecular and materials science

Keith T. Butler, Daniel W. Davies, Hugh Cartwright, Olexandr Isayev & Aron Walsh

Nature **559**, 547–555 (2018) | Download Citation

The Applications of Machine Learning in Biology

BY RAGOTHAMAN YENNAMALLI ON MARCH 15, 2019

ARTIFICIAL INTELLIGENCE, DATA SCIENCE, HEALTHCARE, RESEARCH

0

Machine learning has several applications in diverse fields, ranging from healthcare to natural language processing. Dr. Ragothanam Yennamalli, a computational biologist and Kolabtree freelancer, examines how machine learning and AI are being applied in biology and genomics.

Bioorganic & Medicinal

Volume 28, Issue 17, 15 September 2018, Pa

Recent applications of machine learning in medicinal chemistry

Jane Panteleev [a] [1], Hua Gao [a] [1], Lei Jia [b] [1]

Get rights and content

Show more

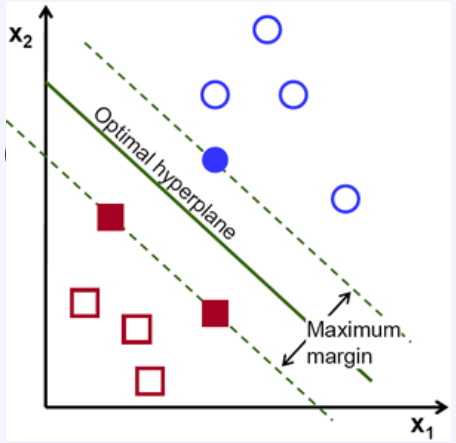
https://doi.org/10.1016/j.bmcl.2018.06.046

Machine learning algorithms

- Regression:
  Linear Regression, Ridge regression, Support Vector Machines, Random Forest,
  Multilayer Neural Networks, Deep Neural Networks, ...

- Classification:
  Logistic Regression, Naive Base, Support Vector Ma
  Random Forest, Multilayer Neural Networks,
  Deep Neural Networks, ...

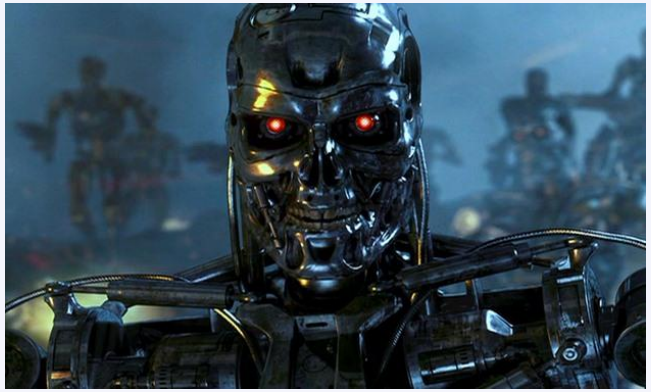
- Clustering:
  k-Means, Hierarchical Clustering, ...



Issues

- Many machine learning/AI projects fail (Gartner claims 85 %)

- Ethics, e.g., Amazon has/had sub-par employees fired by an AI automatically



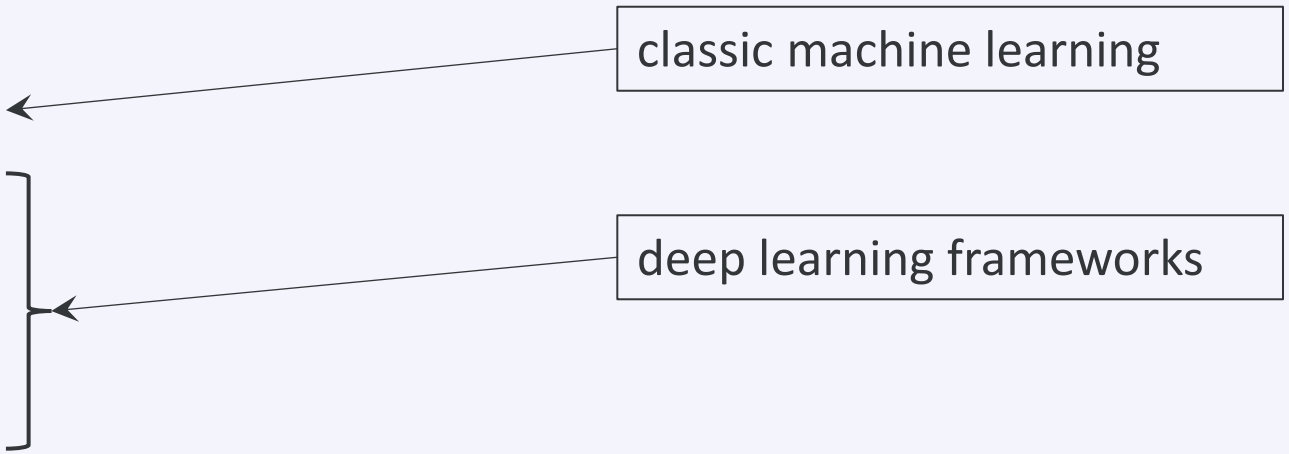
Reasons for failure

- Asking the wrong question
- Trying to solve the wrong problem
- Not having enough data
- Not having the right data
- Having too much data
- Hiring the wrong people
- Using the wrong tools
- Not having the right model
- Not having the right yardstick

Frameworks

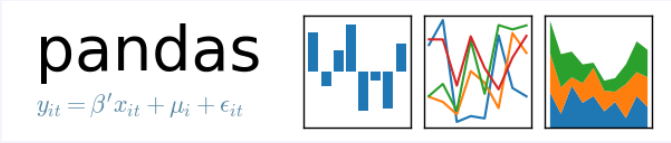
- Programming languages
  - Python
  - R
  - C++
  - …
- Many libraries
  - scikit-learn
  - PyTorch
  - TensorFlow
  - Keras
  - …

Fast-evolving ecosystem!

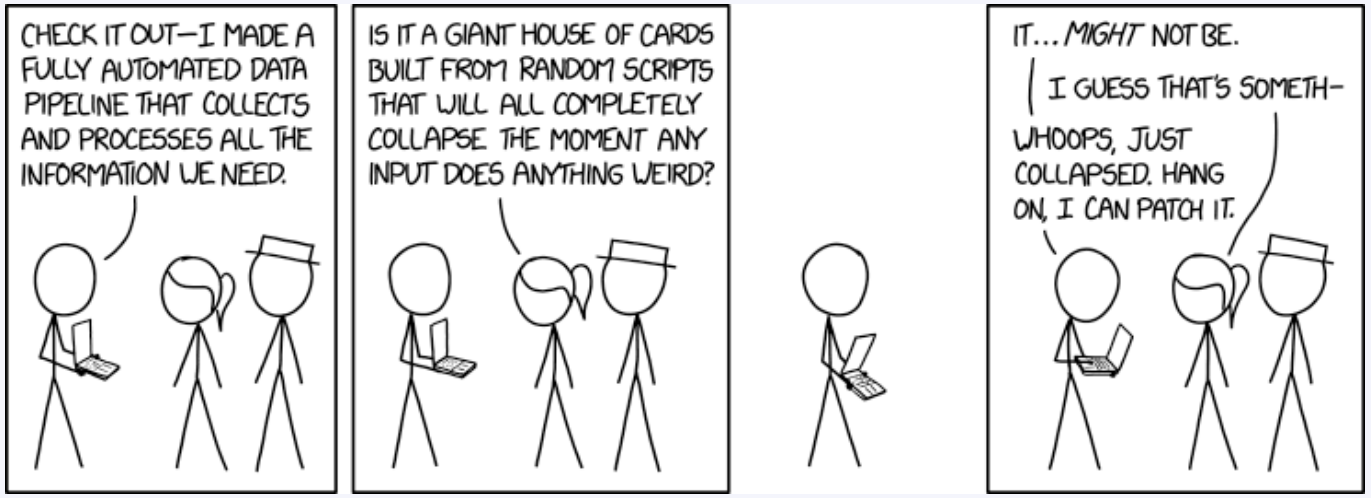


Data pipelines

- Data ingestion
  - CSV/JSON/XML/H5 files, RDBMS, NoSQL, HTTP,...
- Data cleaning
  - outliers/invalid values? → filter
  - missing values? → impute
- Data transformation
  - scaling/normalization

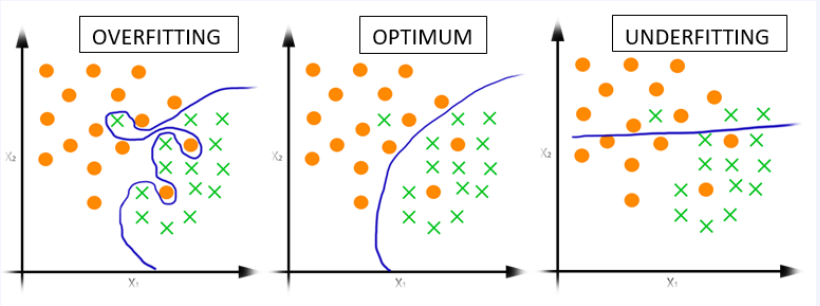


Must be done systematically



Supervised learning: methodology

- Select model, e.g., random forest, (deep) neural network, ...
- Train model, i.e., determine parameters
  - Data: input + output
    - training data → determine model parameters
    - validation data → yardstick to avoid overfitting
- Test model
  - Data: input + output
    - testing data → final scoring of the model
- Production
  - Data: input → predict output



Experiment with underfitting and overfitting:
`010_underfitting_overfitting.ipynb`

https://github.com/gjbex/Python-for-machine-learning/tree/master/source-code/scikit-learn

SCIKIT-LEARN

scikit-learn

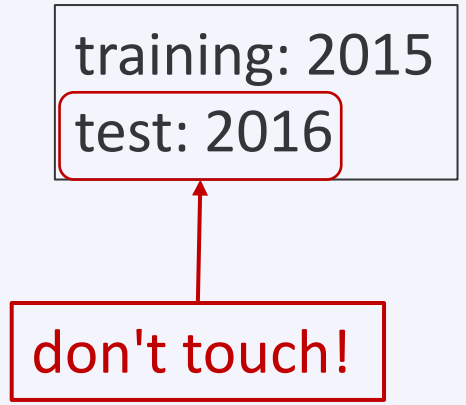
- Nice end-to-end framework
  - data exploration (+ pandas + holoviews)
  - data preprocessing (+ pandas)
    - cleaning/missing values
    - normalization
  - training
  - testing
  - application
- "Classic" machine learning only
- https://scikit-learn.org/stable/

Machine learning tasks

- Supervised learning
  - regression: predict numerical values
  - classification: predict categorical values, i.e., labels
- Unsupervised learning
  - clustering: group data according to "distance"

Data set

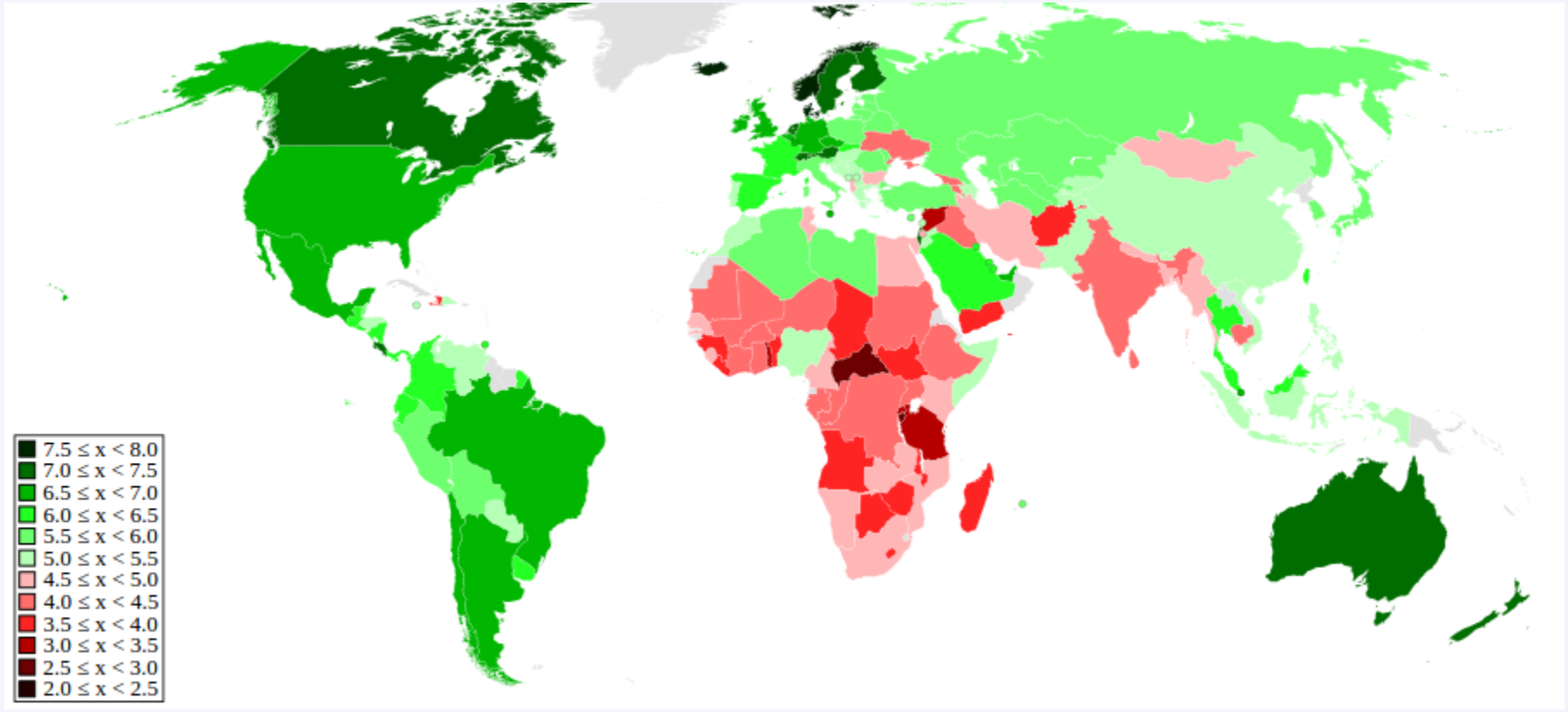
- World happiness index 2015 & 2016 (http://worldhappiness.report/)
- Happiness score for country based on
  - economic factors (GDP)
  - family situation (social network)
  - health care (life expectancy)
  - freedom
  - trust (government corruption)
  - generosity
  - dystopia residual
- Geographical region, e.g.,
  - Western Europe, Southern Asia,…



quantitative data

categorical data

World happiness



https://en.wikipedia.org/wiki/World_Happiness_Report

Task 1

- Given
  - economy
  - family
  - health
  - freedom
  - trust
  - generosity
  - dystopia residual
  - region
- Predict happiness score

Regression

Let's peek…

```
In [4]: data_2015.describe()
```

Out[4]:

	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Generc
count	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	158.000000	158.00(
mean	79.493671	5.375734	0.047885	0.846137	0.991046	0.630259	0.428615	0.143422	0.23
std	45.754363	1.145010	0.017146	0.403121	0.272369	0.247078	0.150693	0.120034	0.12(
min	1.000000	2.839000	0.018480	0.000000	0.000000	0.000000	0.000000	0.000000	0.00(
25%	40.250000	4.526000	0.037268	0.545808	0.856823	0.439185	0.328330	0.061675	0.15(
50%	79.500000	5.232500	0.043940	0.910245	1.029510	0.696705	0.435515	0.107220	0.21(
75%	118.750000	6.243750	0.052300	1.158448	1.214405	0.811013	0.549092	0.180255	0.30(
max	158.000000	7.587000	0.136930	1.690420	1.402230	1.025250	0.669730	0.551910	0.79(

Rescaling

Missing values?

```
In [5]: data_2015.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 158 entries, 0 to 157
Data columns (total 12 columns):
Country                          158 nc
Region                           158 nc
Happiness Rank                   158 nc
Happiness Score                  158 nc
Standard Error                   158 nc
Economy (GDP per Capita)         158 nc
Family                           158 nc
Health (Life Expectancy)         158 nc
Freedom                          158 nc
Trust (Government Corruption)    158 nc
Generosity                       158 nc
Dystopia Residual                158 nc
dtypes: float64(9), int64(1), object(2)
memory usage: 14.9+ KB
```

```
In [63]: data_2015.count()
Out[63]: Country                          158
         Region                           158
         Happiness Rank                   158
         Happiness Score                  158
         Standard Error                   158
         Economy (GDP per Capita)         158
         Family                           158
         Health (Life Expectancy)         158
         Freedom                          158
         Trust (Government Corruption)    158
         Generosity                       158
         Dystopia Residual                158
         dtype: int64
```

No NaNs, otherwise, use `Imputer`

Extracting data

- Part of machine learning pipeline
  - fit + transform
- Extracting columns from pandas data frame
- Transform data
  - scale numerical features to [0, 1]
  - one-hot encoding for regions

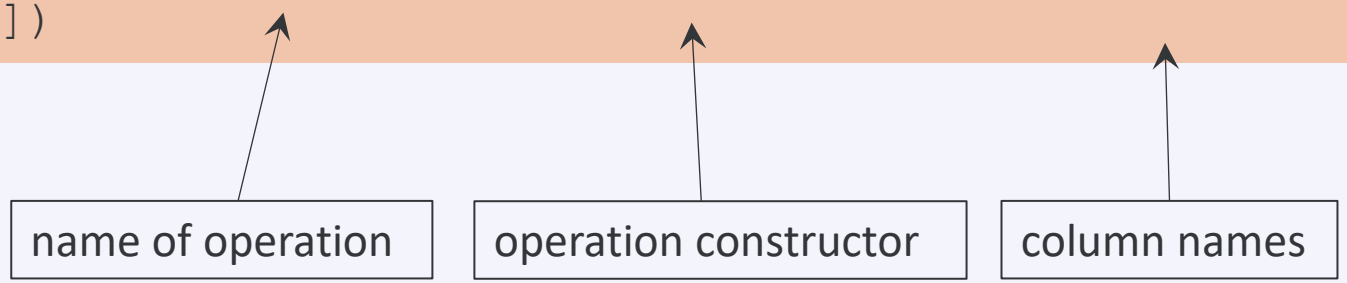
Numerical attributes pipeline

- Extract, then scale

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import MinMaxScaler

num_attr_names = data_2015.columns[5:13]

num_attrs_transformer = ColumnTransformer([
    ('minimax', MinMaxScaler(), num_attr_names)
])
```



Categorical attribute pipeline

- Extract, then create one-hot attributes

Country	Region
Belgium	Western Europe
Canada	North America
Germany	Western Europe
…	…

Country	Western Europe	Northren America	…
Belgium	1	0	…
Canada	0	1	…
Germany	1	0	…
…	…	…	…

```
from sklearn.preprocessing import OneHotEncoder

region_transformer = ColumnTransformer([
    ('one_hot_encoder', OneHotEncoder(categories='auto'),
     ['Region'])
])
```

Combining pipelines & execution

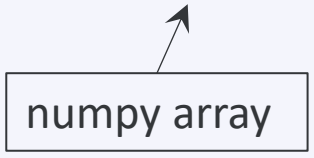
- Both pipelines must be executed, results combined

```
from sklearn.pipeline import FeatureUnion

preparation_pipeline = FeatureUnion(transformer_list=[
    ('region_attr', region_attr_transformer),
    ('num_attrs', num_attrs_transformer),
])
```

- Run data through pipeline

```
train_data = preparation_pipeline.fit_transform(data_2015)
```



Ready to start training!

Training & prediction

- Create learning algorithm

```
from sklearn.linear_model import Ridge

ridge_regr = Ridge(alpha=0.5, fit_intercept=False)
```

hyperparameters

- Train

```
X = train_data
Y = np.array(data_2015['Happiness Score'])
ridge_regr.fit(X, Y);
```

???

- Predict

```
Y_ridge_regr = ridge_regr.predict(X)
```

Score & errors

- Score

```
ridge_regr.score(X, Y)
```

→

```
0.9984
```

- Better: cross validation

```
scores = cross_val_score(
       ridge_regr, X, Y,
        scoring='neg_mean_squared_error',
        cv=10
)
```

10-fold

```
np.sqrt(-scores)
```

→

```
0.1954,  0.0215,
0.0579,  0.0478,
0.0551,  0.0649,
0.0591,  0.0451,
0.0729,  0.0803
```

Fine tuning

- Define hyper parameter search space

```
grid_params = [
    {'alpha': [0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5],
     'fit_intercept': [True, False]},
]
```

- Define grid searcher

```
grid_search = GridSearchCV(ridge_regr, grid_params, cv=10,
                           scoring='neg_mean_squared_error')
```

- Search

```
grid_search.fit(X, Y)
```

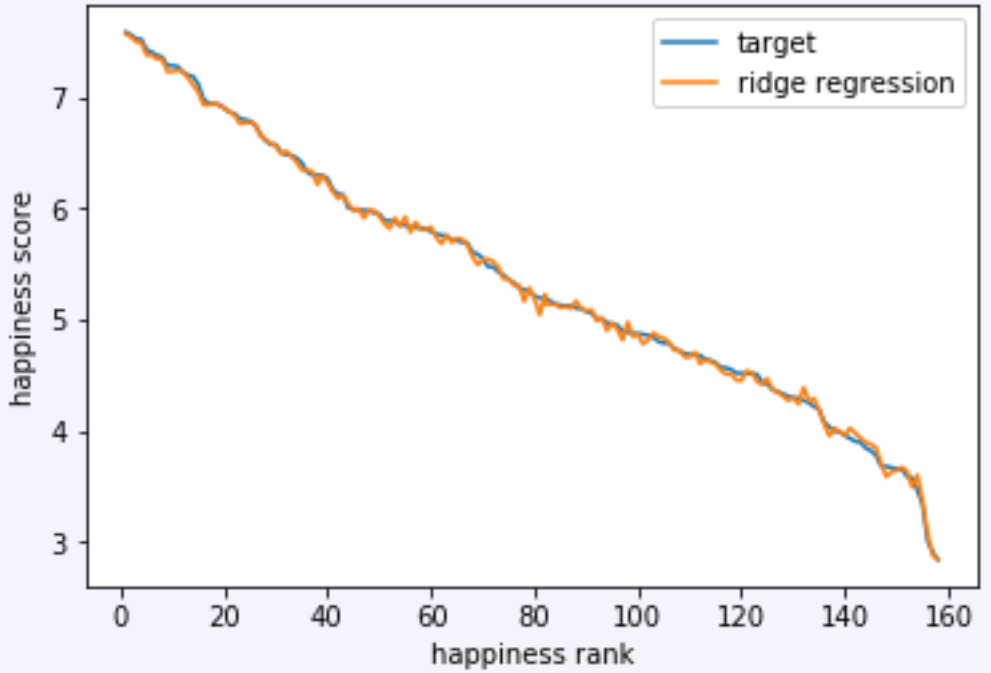
- Best hyper parameters

```
grid_search.best_params_
```

→

```
{'alpha': 0.0005, 'fit_intercept': True}
```

Training result



Testing the model

- Use pipeline

```
test_data = preparation_pipeline.transform(data_2016)
```

Note: only `transform`, no `fit_transform`!

- Predict

```
Y_test = ridge_regr.predict(X_test)
```

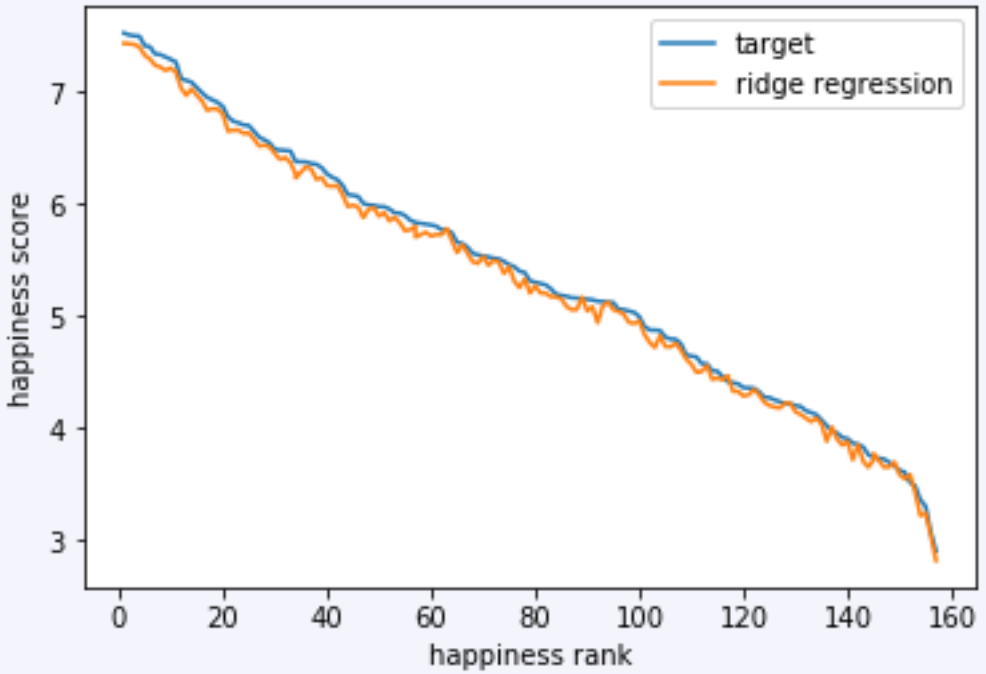
- Score

```
ridge_regr.score(X_test, Y_test)
```

→

```
0.9950
```

Test result


target
ridge regression
happiness score
happiness rank
7
6
5
4
3
0
20
40
60
80
100
120
140
160

Task 2

- Given
  - economy
  - family
  - health
  - freedom
  - trust
  - generosity
  - dystopia residual
- Predict whether country in Western Europe

Classification

Data preparation

- Numerical pipeline is same as in task 1
  - select from pandas `DataFrame`
  - scale

```
prepared_data = num_attrs_pipeline.fit_transform(data_2015)
```

- Add column for class
  - `True` if in Western Europe, `False` otherwise

```
data_2015['Western Europe'] = (data_2015['Region'] ==
                               'Western Europe')
```

Training & scoring

- Create learning algorithm

```
from sklearn.naive_bayes import GaussianNB

nb = GaussianNB()
```

attributes: approx. Gaussian distr.

- Train

```
nb.fit(prepared_data, data_2015['Western Europe'])
```

- Scoring

```
nb.score(prepared_data, data_2015['Western Europe'])
```

→ `0.9367`

Seems reasonable

Testing the model

- Use pipeline

```
test_data = num_attrs_pipeline.transform(data_2016)
```

Note: only `transform`, no `fit_transform`!

- Predict

```
Y_test = nb.predict(test_data)
```

- Score

```
nb.score(test_data, Y_test)
```

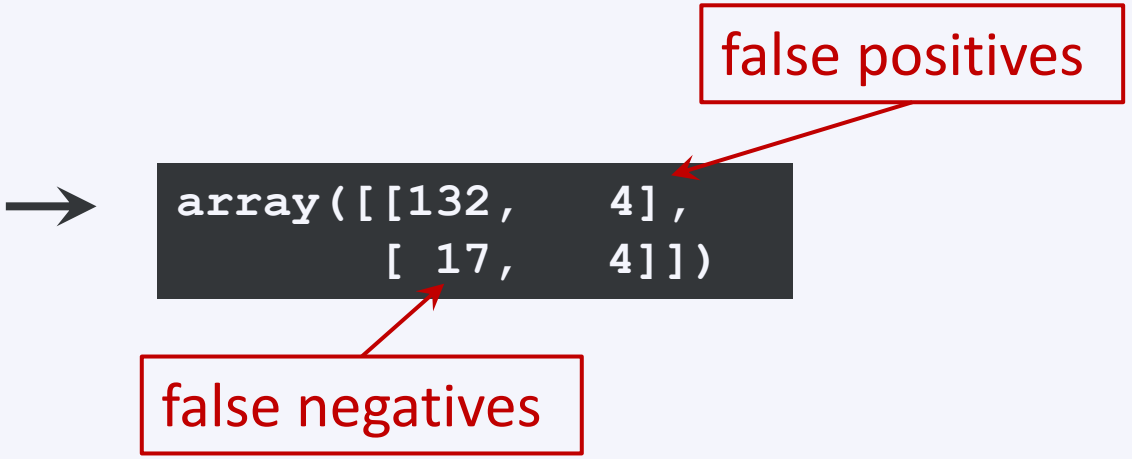
→ `0.8662`

Not so good

Actually…

- Compute confusion matrix

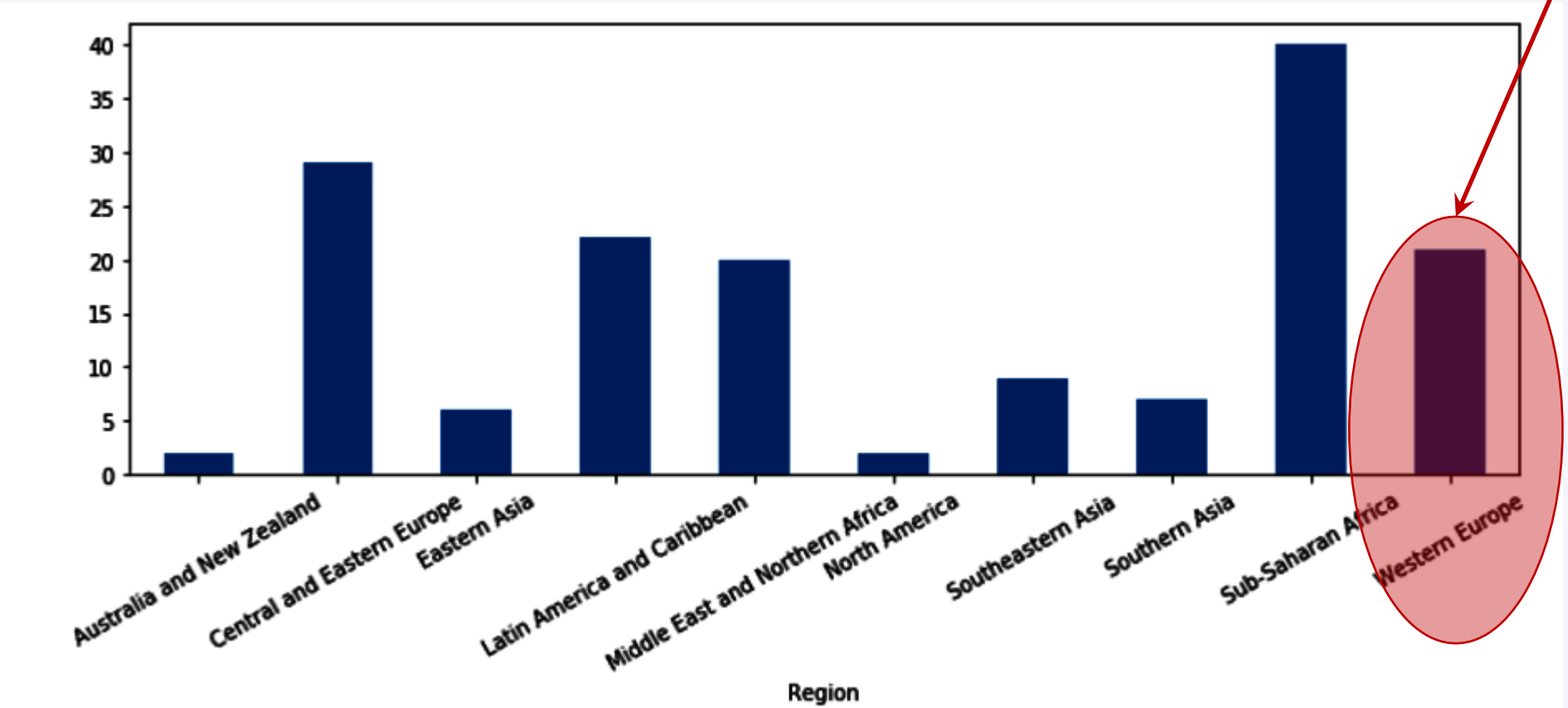
```
confusion_matrix(data_2016['Western Europe'],
                 nb.predict(test_data))
```



Massive exit from Western Europe: *17* countries just left!

Data set properties

small class, many detractors



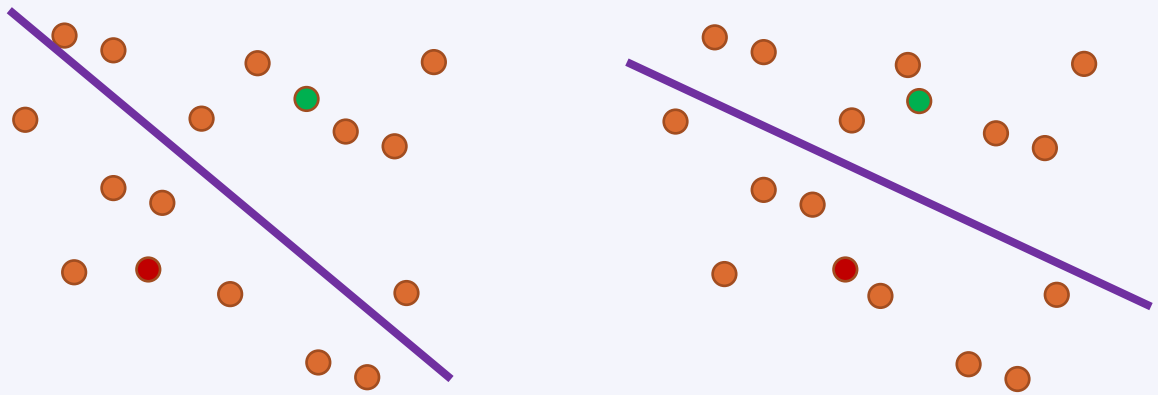
Task 3

- Given
  - economy
  - family
  - health
  - freedom
  - trust
  - generosity
  - dystopia residual
- Find countries that are "close"

Clustering

k-means algorithm

- Start with *k* means, i.e., seed points
- Repeat
  - Assign data points to nearest mean value
  - Update by recomputing mean values
- Stop when no more change



Data preparation & Training

- Data preprocessing same as task 2
- Create learning algorithm

```
from sklearn.cluster import KMeans
k_means = KMeans(n_clusters=3)
```

How many clusters?

- Train

```
k_means.fit(prepared_data)
```

- Add cluster label to data set

```
data_2015['Cluster'] = 0
for i in range(3):
    data_2015.loc[clusterer.labels_ == i, 'Cluster'] = i
```

Examine clusters

```
data_2015.boxplot(by='Cluster', column=num_attr_names)
```


```
data_2015.loc[:, ('Cluster', 'Country')].groupby('Cluster').count()
```

Cluster	Country
0	77
1	50
2	31

And the winners are…

- Switzerland
- Iceland
- Denmark
- Norway
- Canada
- Finland
- Netherlands
- Sweden
- New Zealand
- Australia
- Austria
- United States
- Luxembourg
- Ireland
- Belgium
- United Arab Emirates
- United Kingdom
- Oman
- Singapore
- Germany
- Qatar
- France
- Uruguay
- Saudi Arabia
- Malta
- Kuwait
- Uzbekistan
- Japan
- Bahrain
- Turkmenistan
- Hong Kong

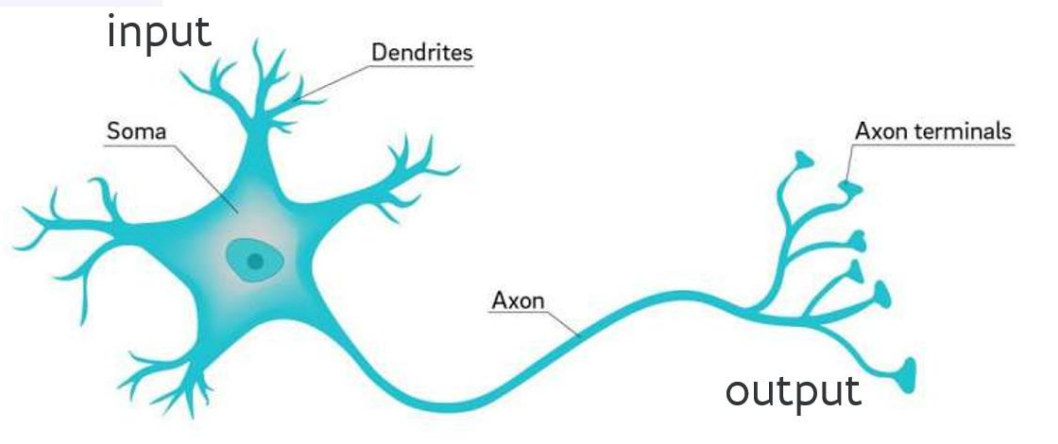
https://github.com/gjbex/Python-for-machine-learning/tree/master/source-code/keras

KERAS

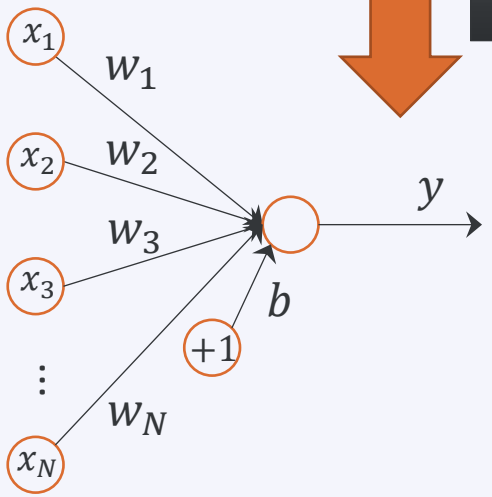
Keras

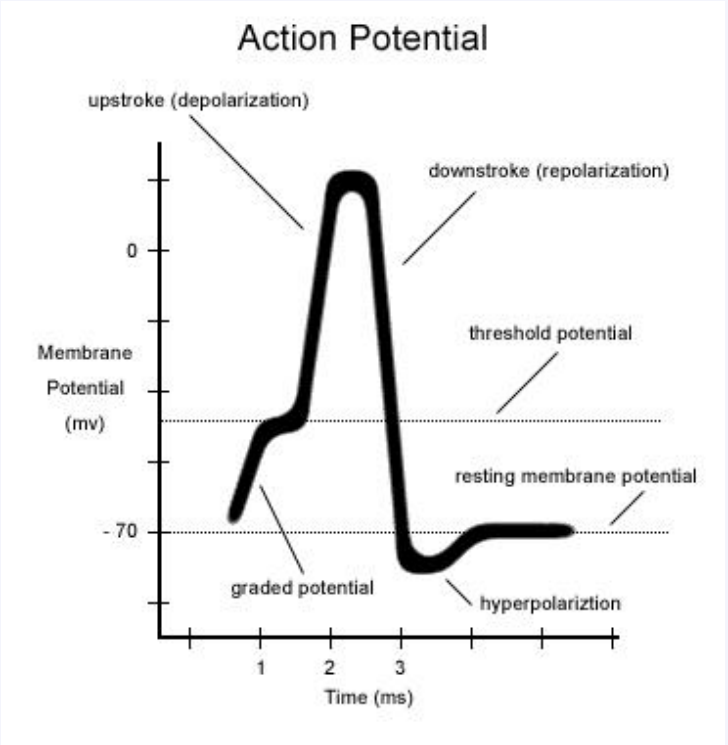
- High-level framework for deep learning
- Interface within TensorFlow package
- Layer types
  - dense
  - convolutional
  - pooling
  - embedding
  - recurrent
  - activation
  - …
- https://keras.io/

From neurons to ANNs

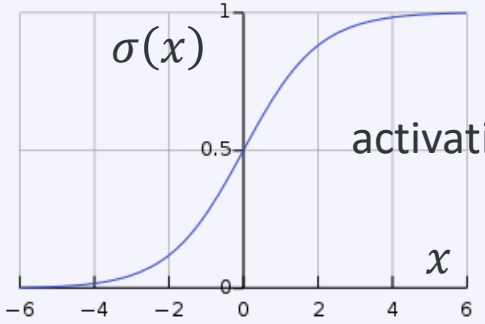


inspiration



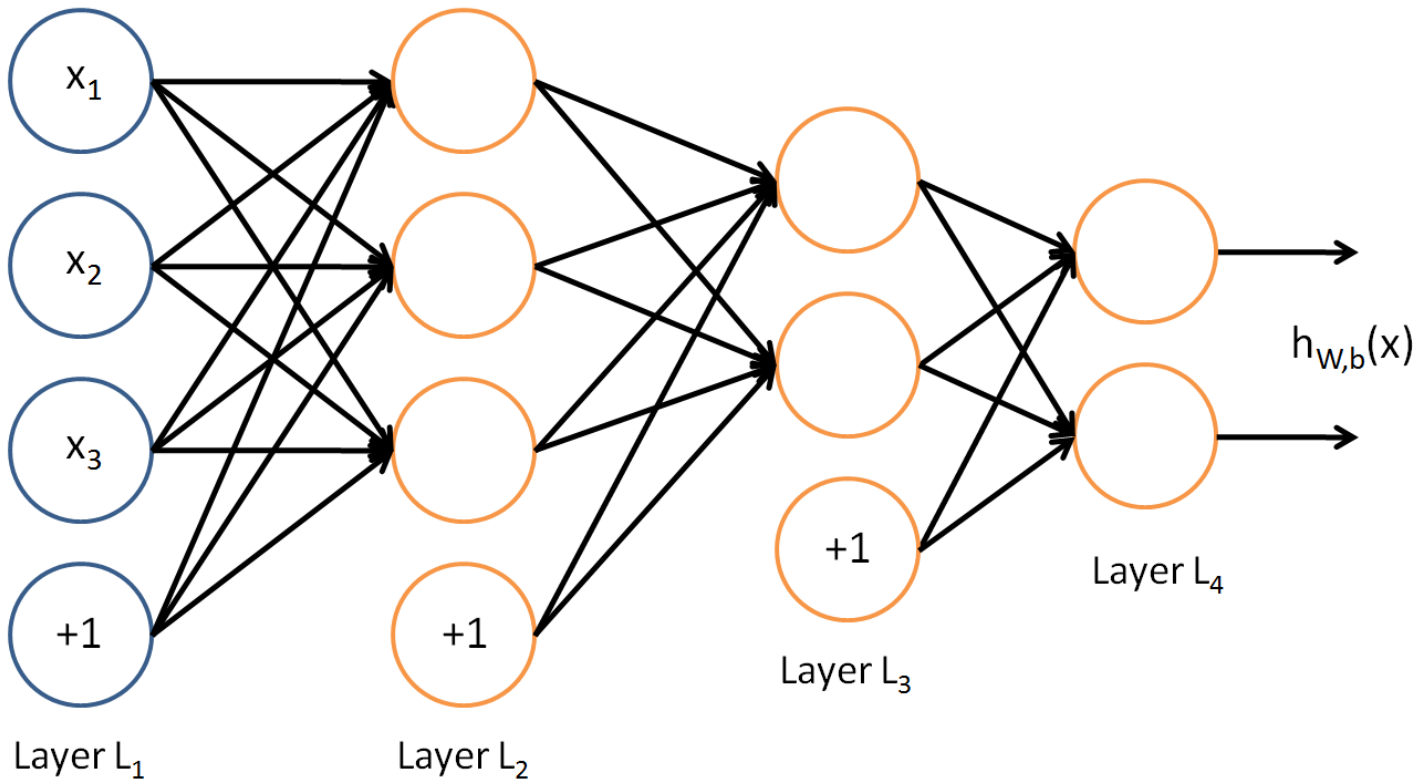


$$y = \sigma\left(\sum_{i=1}^{N} w_i x_i + b\right)$$



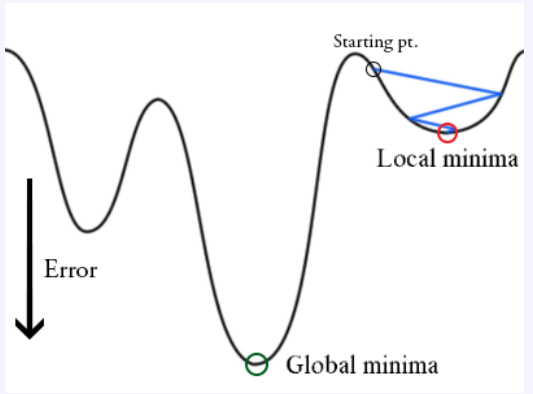
activation function

Multilayer network



How to determine weights?

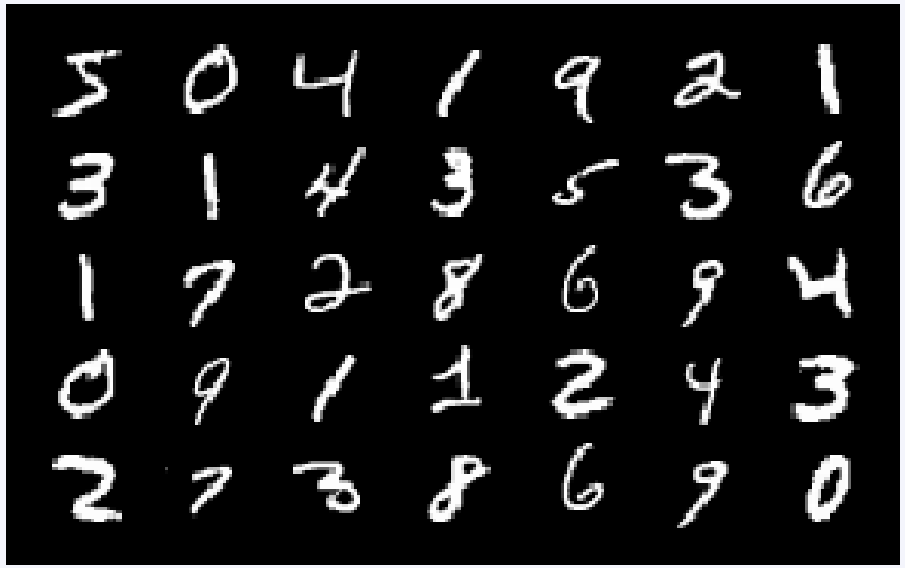
Training: backpropagation



- Initialize weights "randomly"
- For all training epochs
  - for all input-output in training set
    - using input, compute output (forward)
    - compare computed output with training output (loss function)
    - adapt weights (backward) to improve output
  - if accuracy is good enough, stop

Task: handwritten digit recognition

- Input data
  - grayscale image
- Output data
  - digit 0, 1, ..., 9
- Training examples
- Test examples

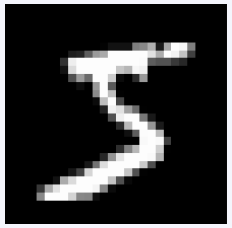


Explore the data: https://github.com/lovnishverma/Python-Getting-Started/blob/main/020_mnist_data_exploration_complete.ipynb

First approach

- Data preprocessing
  - Input data as 1D array
  - output data as array with one-hot encoding

- Model: multilayer perceptron
  - 784 inputs
  - dense hidden layer with 512 units
  - ReLU activation function
  - dense layer with 512 units
  - ReLU activation function
  - dense layer with 10 units
  - SoftMax activation function



```
array([ 0.0, 0.0,..., 0.951, 0.533,..., 0.0, 0.0], dtype=float32)
```

5

```
array([ 0, 0, 0, 0, 0, 1, 0, 0, 0, 0], dtype=uint8)
```

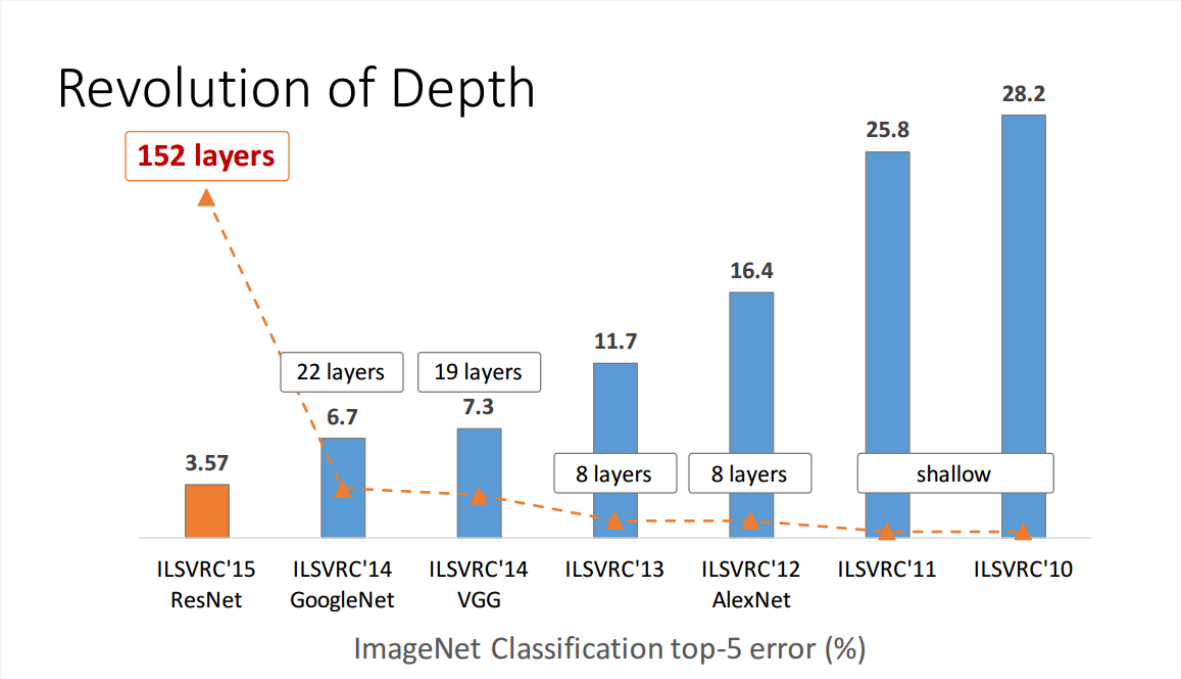
Activation functions: https://github.com/lovnishverma/Python-Getting-Started/blob/main/`030_activation_functions_complete.ipynb`

Multilayer perceptron: https://github.com/lovnishverma/Python-Getting-Started/blob/main/ `040_mnist_mlp_complete.ipynb`

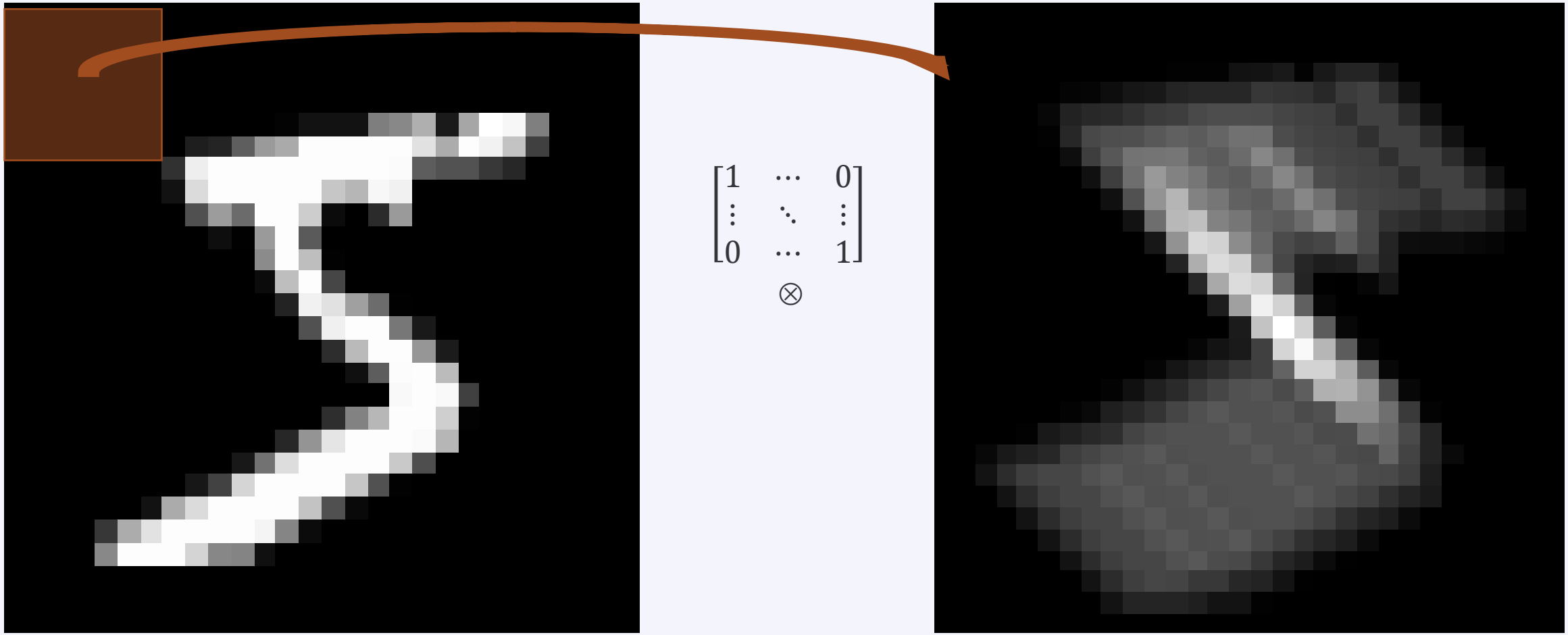
Deep neural networks

- Many layers
- Features are *learned*, not given
- Low-level features combined into high-level features

- Special types of layers
  - convolutional
  - drop-out
  - recurrent
  - …



Convolutional neural networks

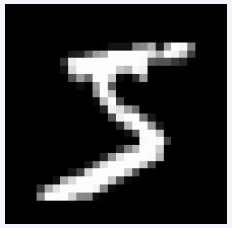


Convolution examples

Convolution: https://github.com/lovnishverma/Python-Getting-Started/blob/main/ `050_convolution_complete.ipynb`

Second approach

- Data preprocessing
  - Input data as 2D array
  - output data as array with one-hot encoding



```
array([[ 0.0, 0.0,..., 0.951, 0.533,..., 0.0, 0.0]], dtype=float32)
```

5

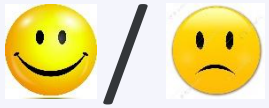
```
array([ 0, 0, 0, 0, 0, 1, 0, 0, 0, 0], dtype=uint8)
```

- Model: convolutional neural network (CNN)
  - 28 × 28 inputs
  - CNN layer with 32 filters 3 × 3
  - ReLU activation function
  - flatten layer
  - dense layer with 10 units
  - SoftMax activation function

Convolutional neural network: https://github.com/lovnishverma/Python-Getting-Started/blob/main/ `060_mnist_cnn_complete.ipynb`

Task: sentiment classification

- Input data
  - movie review (English)
- Output data



- Training examples
- Test examples

```
<start> this film was just brilliant casting location
scenery story direction everyone's really suited the part
they played and you could just imagine being there Robert
redford's is an amazing actor and now the same being director
norman's father came from the same scottish island as myself
so i loved the fact there was a real connection with this
film the witty remarks throughout the film were great it was
just brilliant so much that i bought the film as soon as it
...
```

Explore the data: https://github.com/lovnishverma/Python-Getting-Started/blob/main/070_imdb_data_exploration_complete.ipynb

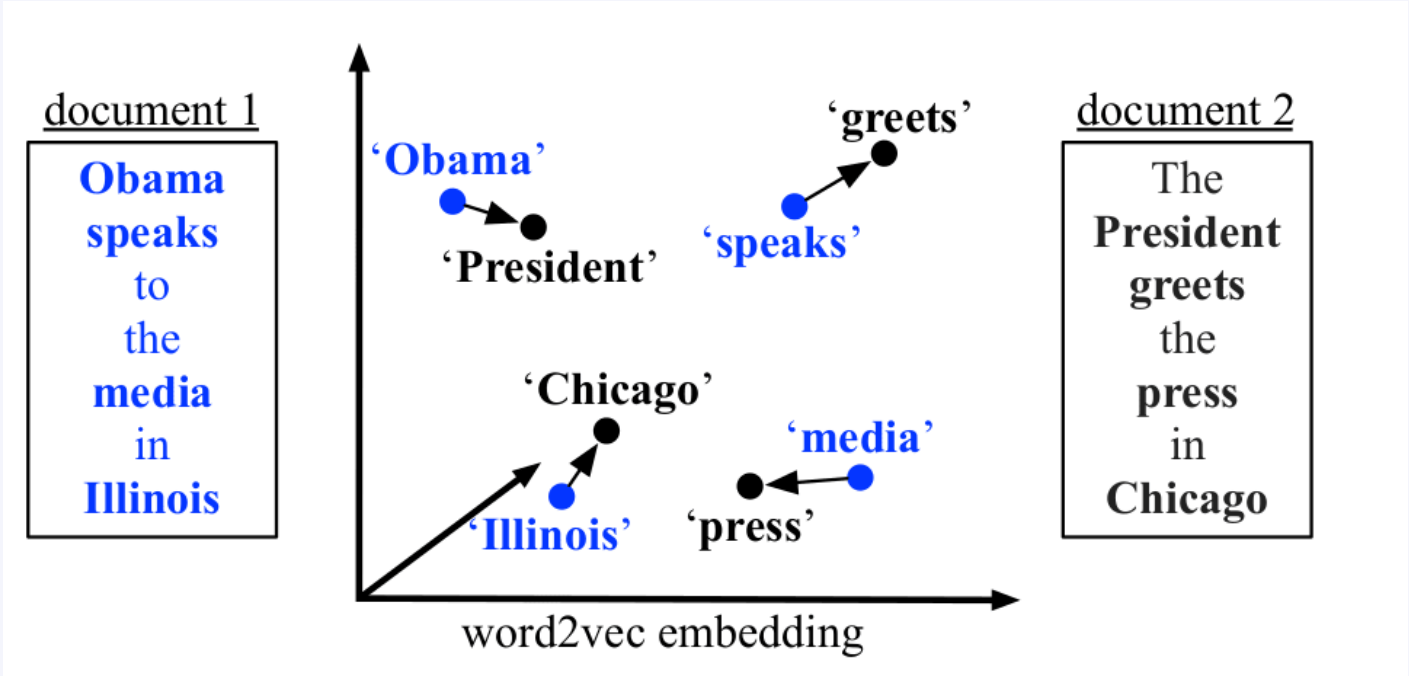
Word embedding

- Represent words as one-hot vectors
  length = vocabulary size

  Issues:
  - unwieldy
  - no semantics

- Word embeddings
  - dense vector
  - vector distance ≈ semantic distance

- Training
  - use context
  - discover relations with surrounding words



How to remember?

Manage history, network learns

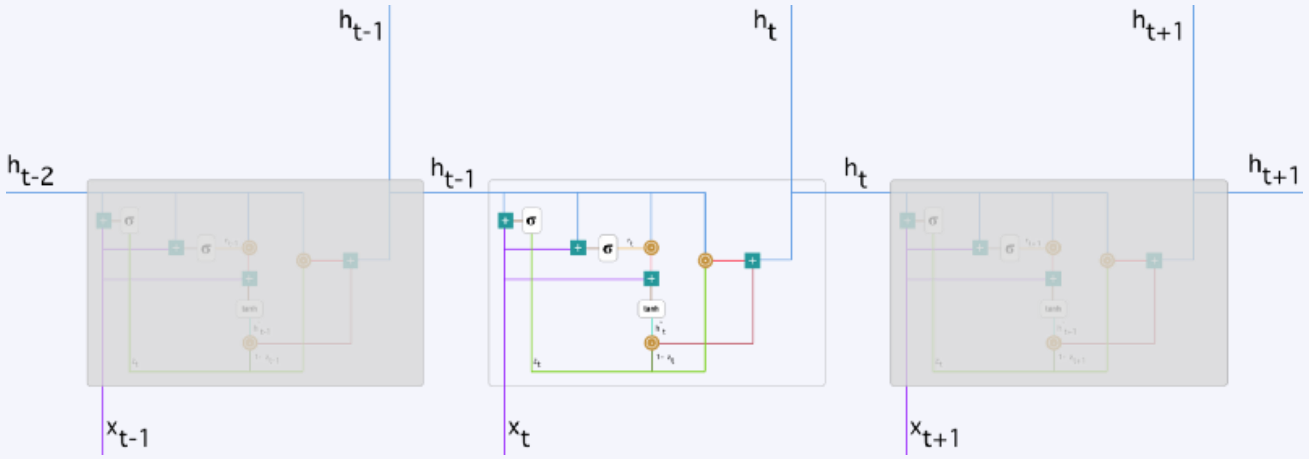
- what to remember
- what to forget

Long-term correlations!

Use, e.g.,

- LSTM (Long Short-Term Memory
- GRU (Gated Recurrent Unit)

Deal with variable length input and/or output



Gated Recurrent Unit (GRU)

- Update gate

$$z_t = \sigma(W_z x_t + U_z h_{t-1})$$

- Reset gate

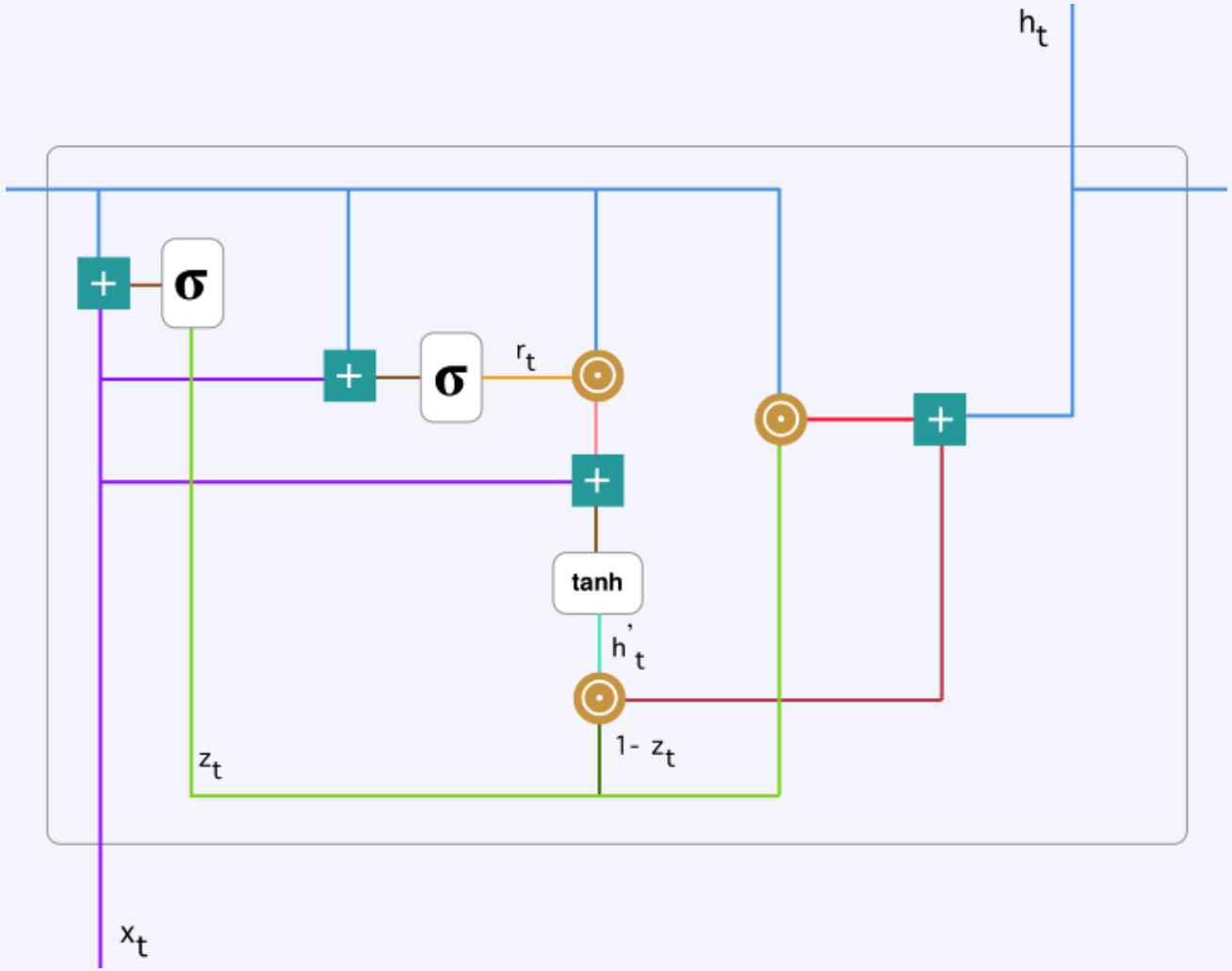
$$r_t = \sigma(W_r x_t + U_r h_{t-1})$$

- Current memory content

$$h'_t = \tanh(W x_t + r_t \odot U h_{t-1})$$

- Final memory/output

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot h'_t$$



Approach

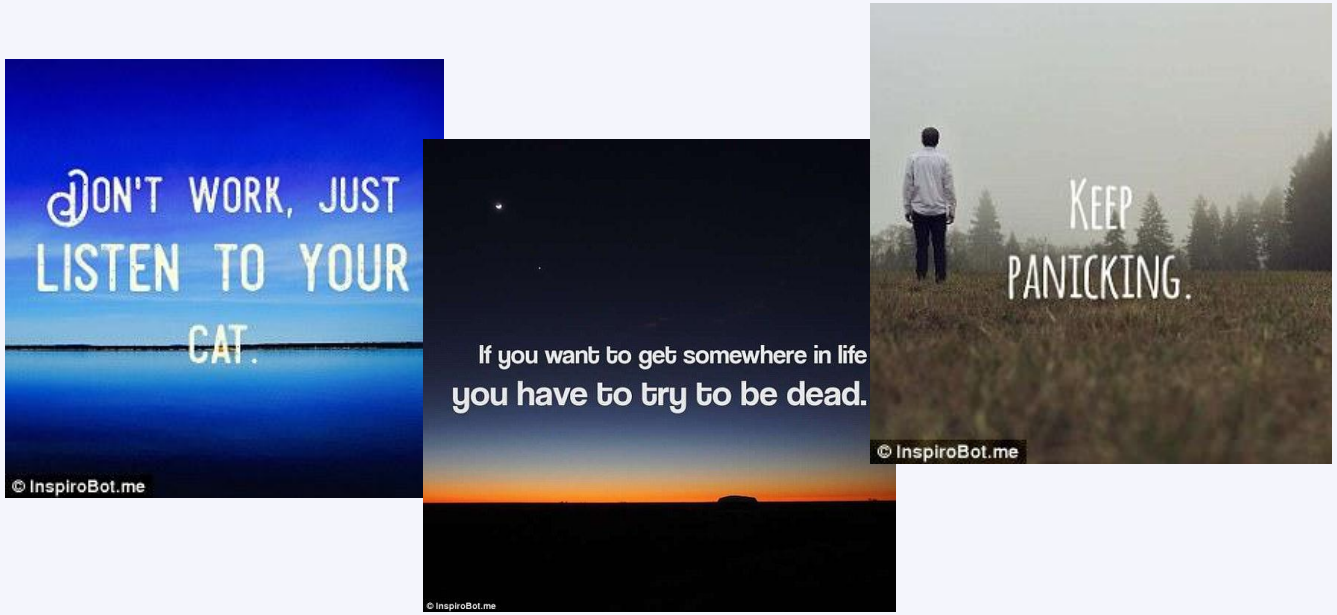
- Data preprocessing
  - Input data as padded array
  - output data as 0 or 1

- Model: recurrent neural network (GRU)
  - 100 inputs
  - embedding layer, 5,000 words, 64 element representation length
  - GRU layer, 64 units
  - dropout layer, rate = 0.5
  - dense layer, 1 output
  - sigmoid activation function

Recurrent neural network: https://github.com/lovnishverma/Python-Getting-Started/blob/main/ `080_imdb_rnn_complete.pynb`

Caveat

- InspiroBot (http://inspirobot.me/)
  - *"I am an artificial intelligence dedicated to generating unlimited amounts of unique inspirational quotes for endless enrichment of pointless human existence".*



Conclusions

- scikit-learn
  - easy to use
  - consistent, elegant API
  - restricted to classic machine learning
- Keras
  - very versatile
  - offers access to TensorFlow backend for low-level
- Consider using pre-trained networks
- Success depends on
  understanding algorithms!
- Success also depends on
  understanding data & domain!
- Many things to explore

Further reading

- *Hands-on machine learning with scikit-learn & TensorFlow*
  Aurélien Géron
  O'Reilly, 2017
- Excellent keras tutorial
  https://github.com/leriomaggio/deep-learning-keras-tensorflow