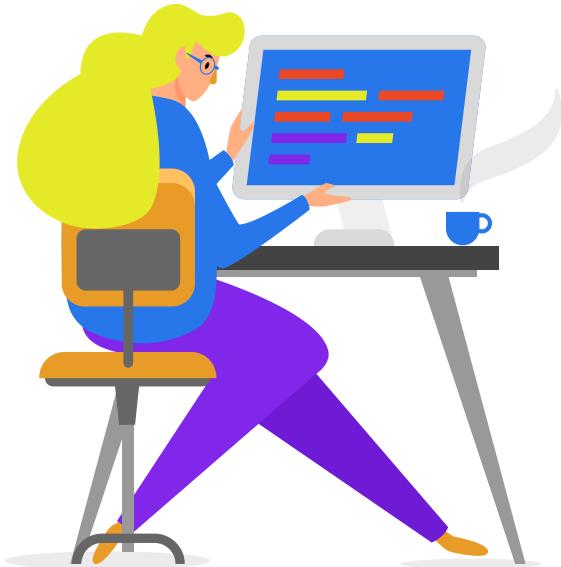


Data Science Machine Learning Python course

Thang Nguyen
Viet Nguyen

<http://www.viet-it.com>

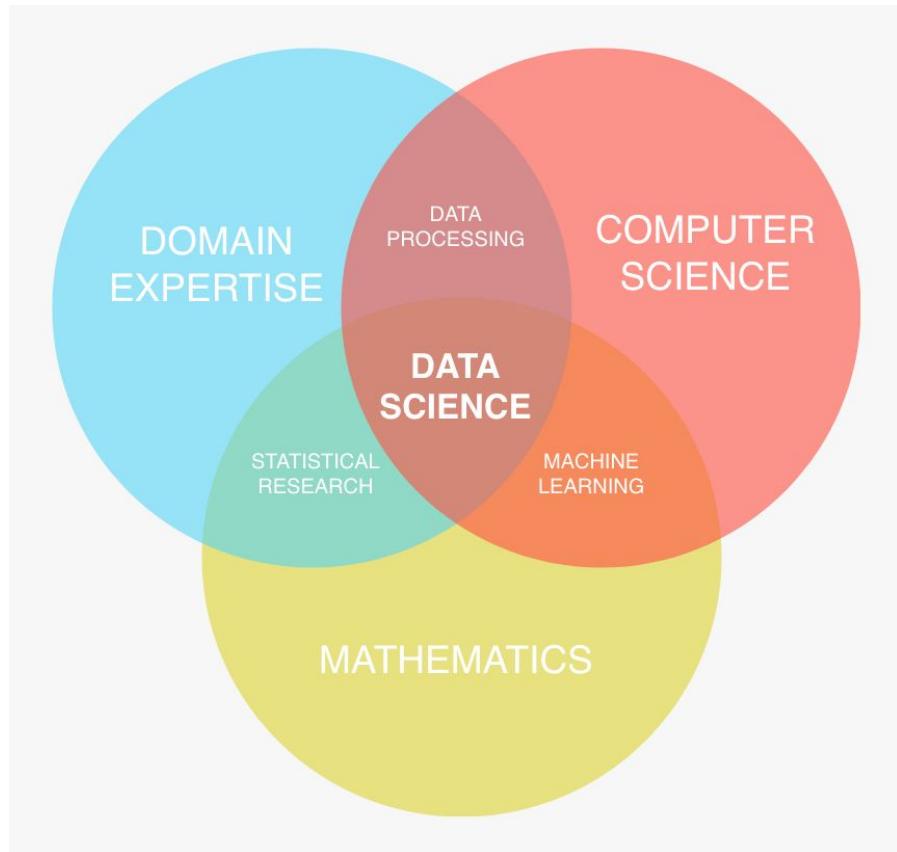
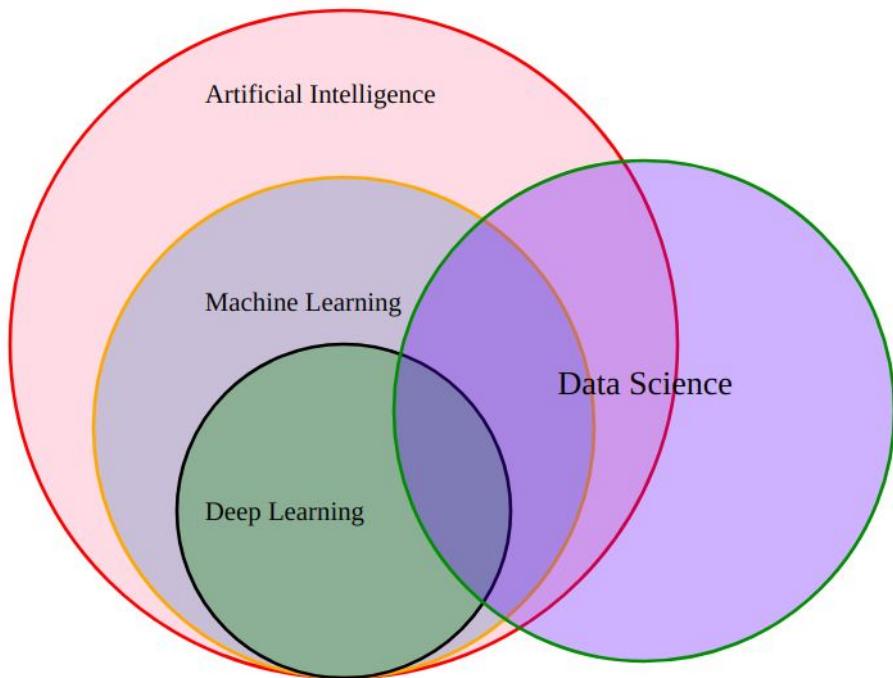
Nội dung



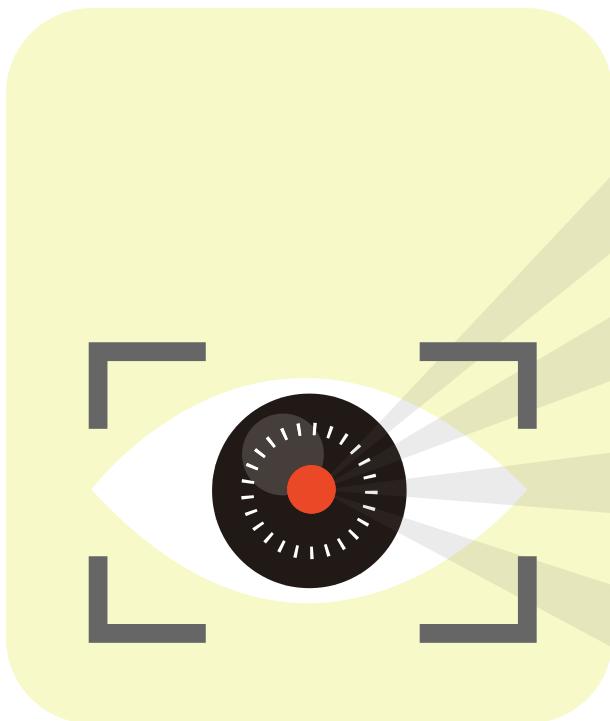
- 01 **Machine Learning**
Tổng quan về Machine Learning
- 02 **Supervised Learning**
Các mô hình cơ bản
- 03 **Unsupervised Learning**
Các mô hình cơ bản
- 04 **Data Visualization**
Trực quan hóa dữ liệu
- 05 **Model evaluation**
Đánh giá mô hình
- 06 **Projects**
Xây dựng mô hình với các datasets

Tổng quan về Machine Learning

Tổng quan về Machine Learning và Data Science



Tổng quan các khái niệm



AI

Đưa ra hành động từ
dữ liệu

Machine Learning

Học hỏi từ dữ liệu

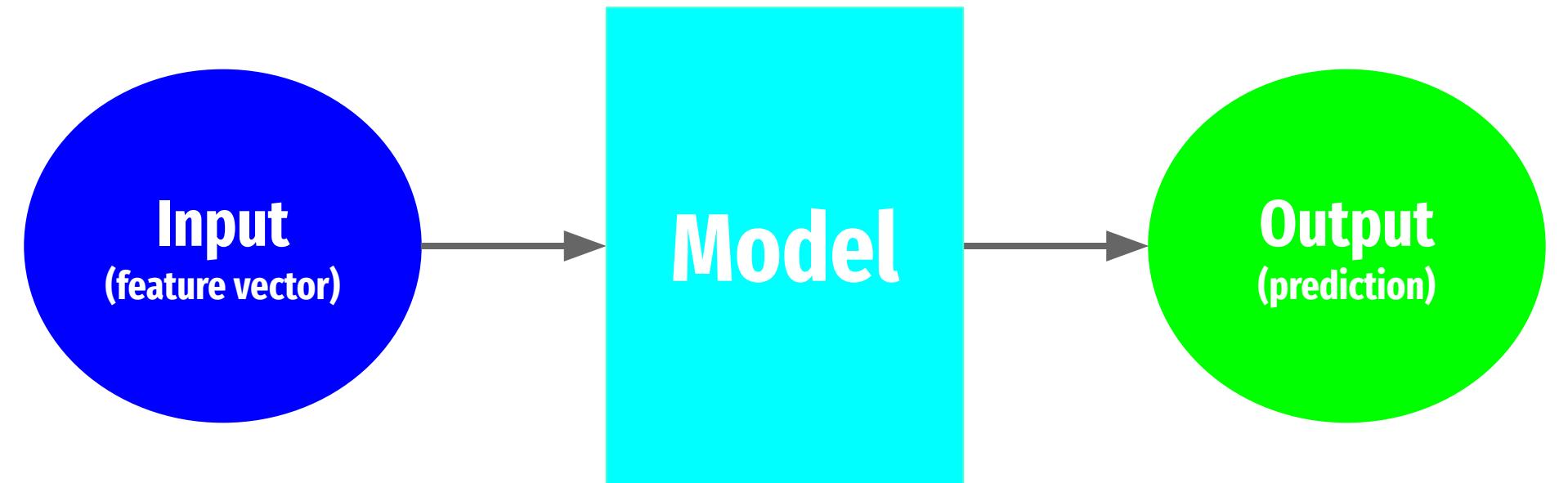
Deep Learning

Học hỏi từ RẤT NHIỀU
dữ liệu

Data Science

Tạo ra giá trị từ dữ liệu

Machine Learning



Dataset (Bộ dữ liệu)

Adult dataset

age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.gain	capital.loss	hours.per.week	native.country	income
90	?	77053	HS-grad	9	Widowed	?	Not-in-family	White	Female	0	4356	40	United-States	<=50K
82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-in-family	White	Female	0	4356	18	United-States	<=50K
66	?	186061	Some-college	10	Widowed	?	Unmarried	Black	Female	0	4356	40	United-States	<=50K
54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	Unmarried	White	Female	0	3900	40	United-States	<=50K
41	Private	264663	Some-college	10	Separated	Prof-specialty	Own-child	White	Female	0	3900	40	United-States	<=50K
34	Private	216864	HS-grad	9	Divorced	Other-service	Unmarried	White	Female	0	3770	45	United-States	<=50K
38	Private	150601	10th	6	Separated	Adm-clerical	Unmarried	White	Male	0	3770	40	United-States	<=50K
74	State-gov	88638	Doctorate	16	Never-married	Prof-specialty	Other-relative	White	Female	0	3683	20	United-States	>50K
68	Federal-gov	422013	HS-grad	9	Divorced	Prof-specialty	Not-in-family	White	Female	0	3683	40	United-States	<=50K
41	Private	70037	Some-college	10	Never-married	Craft-repair	Unmarried	White	Male	0	3004	60	?	>50K
45	Private	172274	Doctorate	16	Divorced	Prof-specialty	Unmarried	Black	Female	0	3004	35	United-States	>50K
38	Self-emp-not-inc	164526	Prof-school	15	Never-married	Prof-specialty	Not-in-family	White	Male	0	2824	45	United-States	>50K
52	Private	129177	Bachelors	13	Widowed	Other-service	Not-in-family	White	Female	0	2824	20	United-States	>50K
32	Private	136204	Masters	14	Separated	Exec-managerial	Not-in-family	White	Male	0	2824	55	United-States	>50K
51	?	172175	Doctorate	16	Never-married	?	Not-in-family	White	Male	0	2824	40	United-States	>50K
46	Private	45363	Prof-school	15	Divorced	Prof-specialty	Not-in-family	White	Male	0	2824	40	United-States	>50K
45	Private	172822	11th	7	Divorced	Transport-moving	Not-in-family	White	Male	0	2824	76	United-States	>50K
57	Private	317847	Masters	14	Divorced	Exec-managerial	Not-in-family	White	Male	0	2824	50	United-States	>50K
22	Private	119592	Assoc-acdm	12	Never-married	Handlers-cleaners	Not-in-family	Black	Male	0	2824	40	?	>50K
34	Private	203034	Bachelors	13	Separated	Sales	Not-in-family	White	Male	0	2824	50	United-States	>50K
37	Private	188774	Bachelors	13	Never-married	Exec-managerial	Not-in-family	White	Male	0	2824	40	United-States	>50K

Feature (thuộc tính)

Numerical (số)

Integer (số nguyên)

- 1, 2, 3
- -2, -1, 0, 1, 2

Float (số thực)

- 0.1, 0.2, 0.3

Categorical (phân loại)

Nominal (định danh)

- Red, green, blue
- US, Vietnam, Thailand

Ordinal (thứ bậc)

- Happy, normal, sad
- XS, S, M, L, XL

Boolean (logic)

- True, False
- Yes, No

Các phương thức học trong Machine Learning

Supervised learning

Classification

- Fraud detection
- Email spam detection
- Diagnostics
- Image classification

Regression

- Risk assessment
- Score prediction

Unsupervised learning

Clustering

- Customer clustering
- Data visualization

Association

- Recommendation System

Reinforcement learning

- Game theory
- Autonomous cars

Supervised learning

Labeled data (dữ liệu có nhãn)

Diabetes dataset

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
6	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
8	183	64	0	0	23.3	0.672	32	1
1	89	66	23	94	28.1	0.167	21	0
0	137	40	35	168	43.1	2.288	33	1
5	116	74	0	0	25.6	0.201	30	0
3	78	50	32	88	31	0.248	26	1
10	115	0	0	0	35.3	0.134	29	0
2	197	70	45	543	30.5	0.158	53	1
8	125	96	0	0	0	0.232	54	1
4	110	92	0	0	37.6	0.191	30	0
10	168	74	0	0	38	0.537	34	1
10	139	80	0	0	27.1	1.441	57	0
1	189	60	23	846	30.1	0.398	59	1
5	166	72	19	175	25.8	0.587	51	1

Labeled data (dữ liệu có nhãn)

Diabetes dataset

sample

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
6	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
8	183	64	0	0	23.3	0.672	32	1
1	89	66	23	99	28.1	0.167	21	0
0	137	40	35	168	43.1	2.288	33	1
5	116	74	0	0	25.6	0.201	30	0
3	78	50	32	88	31	0.248	26	1
10	115	0	0	0	35.3	0.134	29	0
2	197	70	45	543	30.5	0.158	53	1
8	125	96	0	0	0	0.232	54	1
4	110	92	0	0	37.6	0.191	30	0
10	168	74	0	0	38	0.537	34	1
10	139	80	0	0	27.1	1.441	57	0
1	189	60	23	846	30.1	0.398	59	1
5	166	72	19	175	25.8	0.587	51	1

Labeled data (dữ liệu có nhãn)

Diabetes dataset

feature

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
6	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
8	183	64	0	0	23.3	0.672	32	1
1	89	66	23	94	28.1	0.167	21	0
0	137	40	35	163	43.1	2.288	33	1
5	116	74	0	0	25.6	0.201	30	0
3	78	50	32	88	31	0.248	26	1
10	115	0	0	0	35.3	0.134	29	0
2	197	70	45	543	30.5	0.158	53	1
8	125	96	0	0	0	0.232	54	1
4	110	92	0	0	37.6	0.191	30	0
10	168	74	0	0	38	0.537	34	1
10	139	80	0	0	27.1	1.441	57	0
1	189	60	23	846	30.1	0.398	59	1
5	166	72	19	175	25.8	0.587	51	1

Labeled data (dữ liệu có nhãn)

Diabetes dataset

label (target)

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
6	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
8	183	64	0	0	23.3	0.672	32	1
1	89	66	23	94	28.1	0.167	21	0
0	137	40	35	168	43.1	2.288	33	1
5	116	74	0	0	25.6	0.201	30	0
3	78	50	32	88	31	0.248	26	1
10	115	0	0	0	35.3	0.134	29	0
2	197	70	45	543	30.5	0.158	53	1
8	125	96	0	0	0	0.232	54	1
4	110	92	0	0	37.6	0.191	30	0
10	168	74	0	0	38	0.537	34	1
10	139	80	0	0	27.1	1.441	57	0
1	189	60	23	846	30.1	0.398	59	1
5	166	72	19	175	25.8	0.587	51	1

Labeled data (dữ liệu có nhãn)

Diabetes dataset

feature vector

corresponding label (target)

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
6	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
8	183	64	0	0	23.3	0.672	32	1
1	89	66	23	94	28.1	0.167	21	0
0	137	40	35	168	43.1	2.288	33	1
5	116	74	0	0	25.6	0.201	30	0
3	78	50	32	88	31	0.248	26	1
10	115	0	0	0	35.3	0.134	29	0
2	197	70	45	543	30.5	0.158	53	1
8	125	96	0	0	0	0.232	54	1
4	110	92	0	0	37.6	0.191	30	0
10	168	74	0	0	38	0.537	34	1
10	139	80	0	0	27.1	1.441	57	0
1	189	60	23	846	30.1	0.398	59	1
5	166	72	19	175	25.8	0.587	51	1

Labeled data (dữ liệu có nhãn)

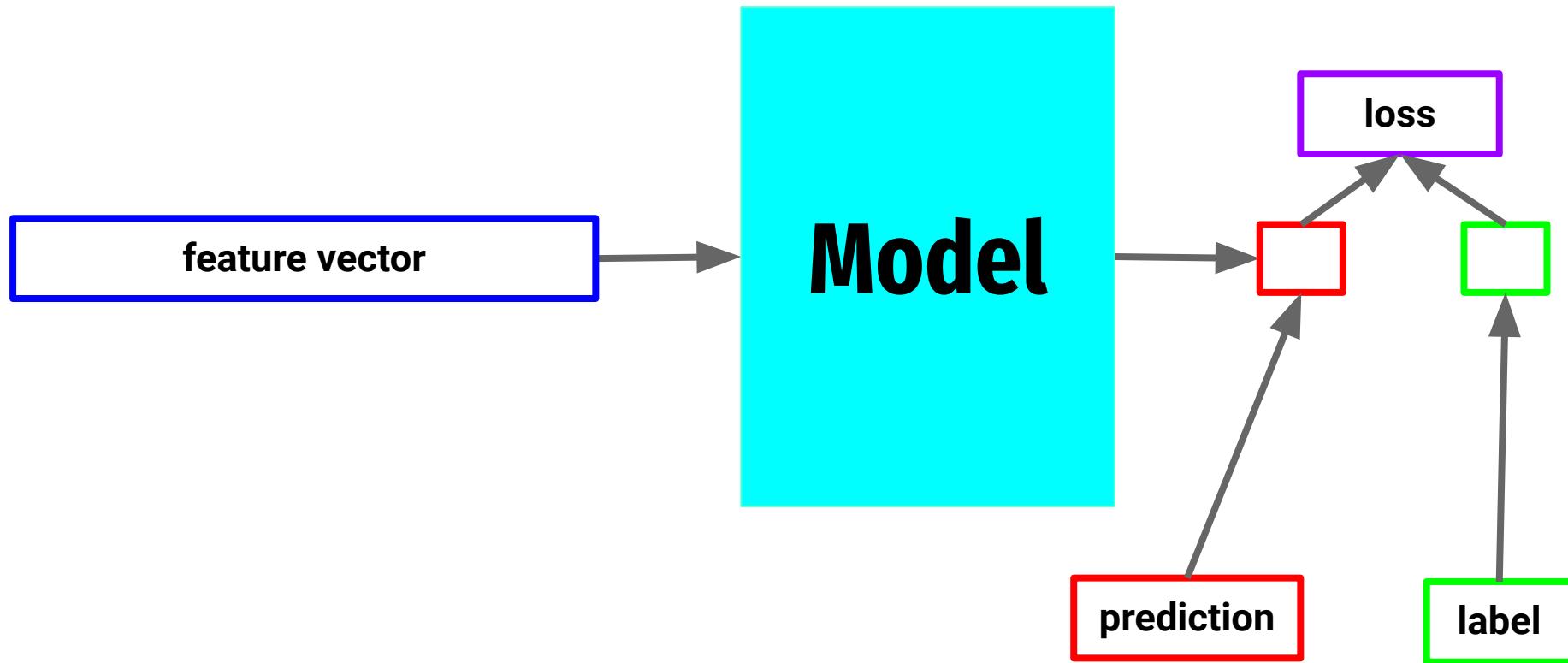
Diabetes dataset

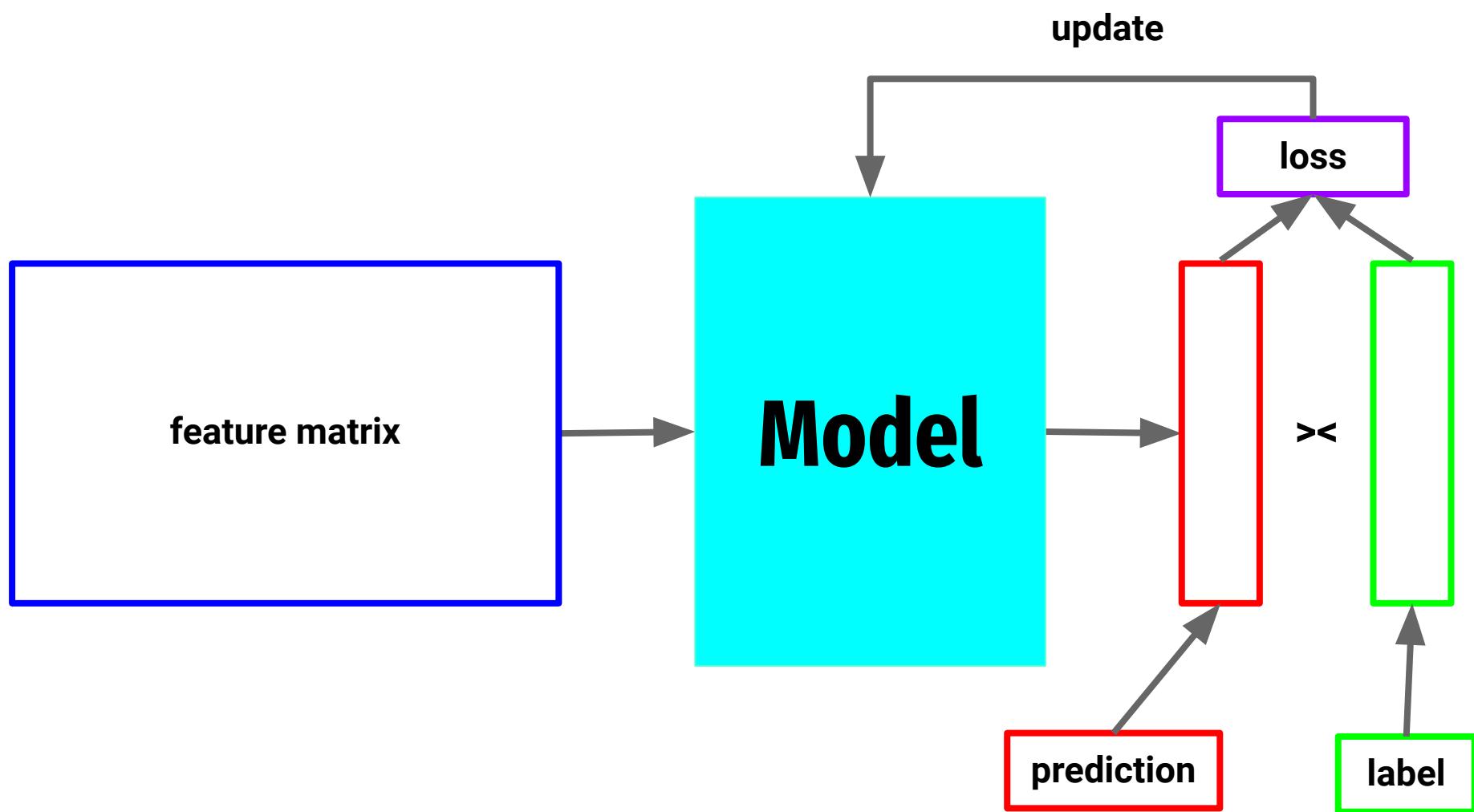
feature matrix (X)

Label/target vector (y)

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
6	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
8	183	64	0	0	23.3	0.672	32	1
1	89	66	23	94	28.1	0.167	21	0
0	137	40	35	168	43.1	2.288	33	1
5	116	74	0	0	25.6	0.201	30	0
3	78	50	32	88	31	0.248	26	1
10	115	0	0	0	35.3	0.134	29	0
2	197	70	45	543	30.5	0.158	53	1
8	125	96	0	0	0	0.232	54	1
4	110	92	0	0	37.6	0.191	30	0
10	168	74	0	0	38	0.537	34	1
10	139	80	0	0	27.1	1.441	57	0
1	189	60	23	846	30.1	0.398	59	1
5	166	72	19	175	25.8	0.587	51	1

Model training (Huấn luyện mô hình)

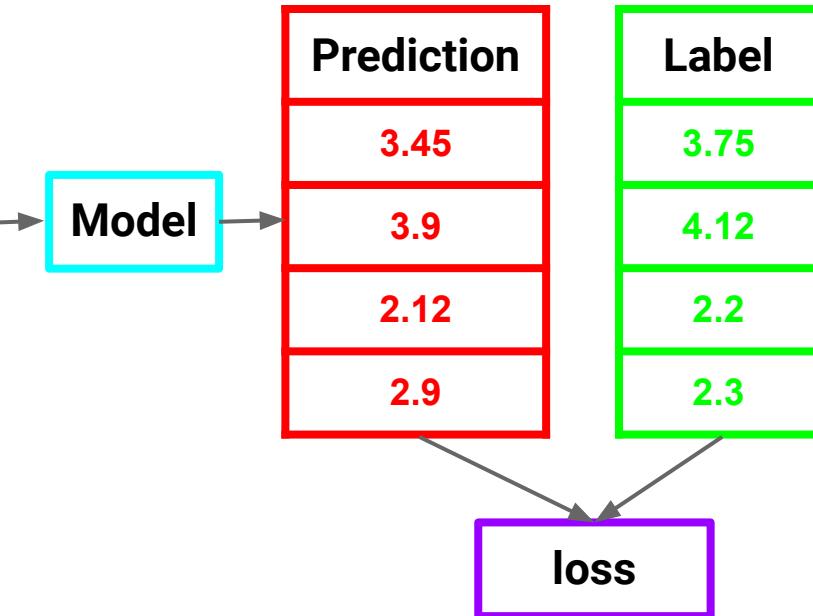




Loss function (Hàm mất mát)

Ví dụ về mô hình dự đoán giá nhà ở Việt Nam (Đơn vị: Tỉ VND)

Diện tích	Số tầng	Thành phố	Trong ngõ
40	6	HCM	No
60	4	Hà Nội	No
120	3	Hải Dương	Yes
50	4	Huế	No



Loss function (Hàm mất mát)

L1 loss

$$\text{Loss} = \sum |y_{\text{prediction}} - y_{\text{actual}}|$$

Các tên gọi:

- Least Absolute Deviations (LAD)
- Absolute Error
- Mean of these Absolute Errors

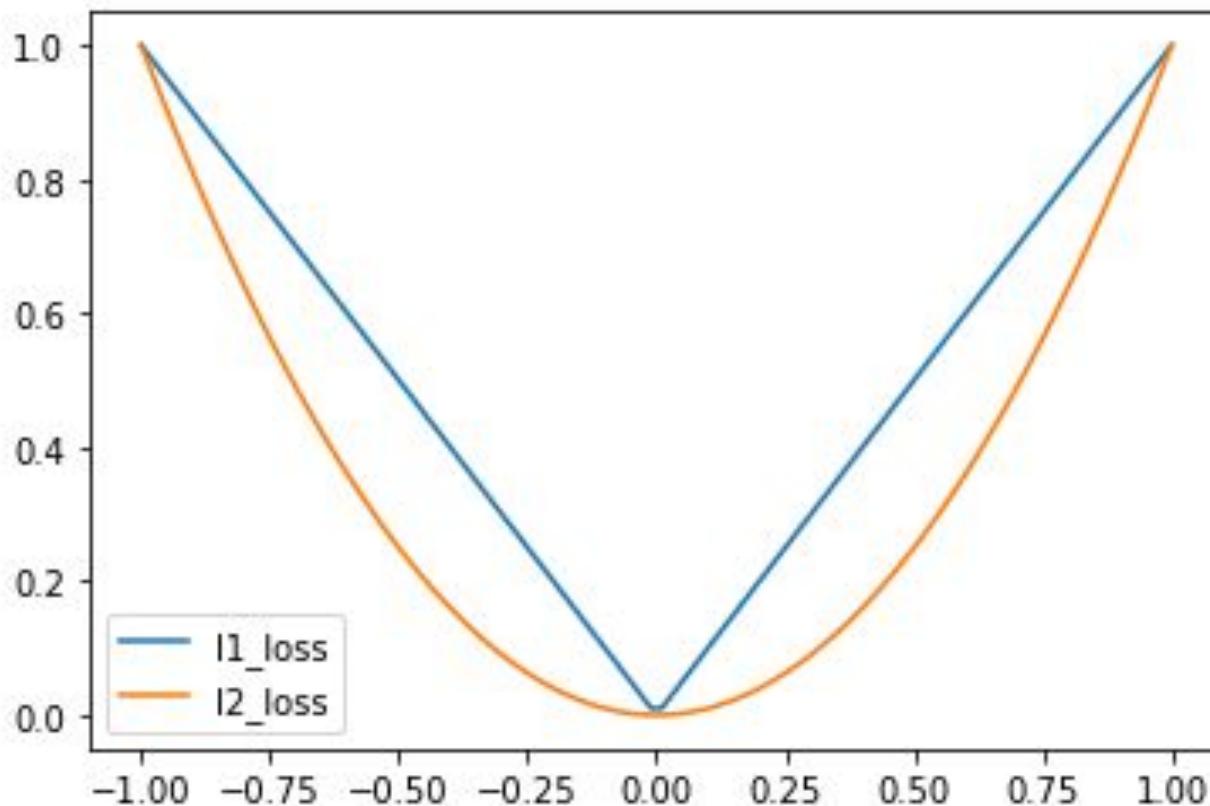
L2 loss

$$\text{Loss} = \sum (y_{\text{prediction}} - y_{\text{actual}})^2$$

Các tên gọi:

- Least Square Errors (LS)
- Squared Error
- Mean of these Squared Errors

Loss function (Hàm mất mát)



Phân chia dataset

Diabetes dataset

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
6	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
8	183	64	0	0	23.3	0.672	32	1
1	89	66	23	94	28.1	0.167	21	0
0	137	40	31	63	31.1	2.288	33	1
5	116	74	0	0	25.6	0.201	30	0
3	78	50	32	88	31	0.248	26	1
10	115	0	0	0	35.3	0.134	29	0
2	197	70	45	543	30.5	0.158	53	1
8	125	96	0	0	0	0.232	54	1
4	110	92	0	0	37.6	0.191	30	0
10	168	74	0	0	38	0.537	34	1
10	139	80	0	0	27.1	1.441	57	0
1	189	60	23	846	30.1	0.398	59	1
5	166	72	19	175	25.8	0.587	51	1

Các bài toán trong Supervised Learning

Classification (phân loại)

Binary Classification

- Fraud detection
- Email spam detection
- Diagnostics

Multiclass Classification

- Animals (cat/dog/horse)
- Fruits (apple/orange/lemon)
- Flowers (rose/daisy/sunflower)

Regression(hồi quy)

- House price estimation
- Stock prediction
- Temperature forecast

Classification

“Is this email spam or not spam?”

To: daniel@mrdourke.com

Hey Daniel,

This deep learning course is incredible! C0ongratu1ations! U win \$1139239230
I can't wait to use what I've learned!

Not spam

To: daniel@mrdourke.com

Hay daniel...

C0ongratu1ations! U win \$1139239230

Spam

“Is this a photo of sushi, steak or pizza?”



Binary classification *(one thing or another)*

“What tags should this article have?”

A screenshot of a Wikipedia article page for "Deep learning". The page content discusses deep learning as a subset of machine learning. Below the main text, there is a sidebar with categories like "Machine learning", "Representation learning", and "Artificial intelligence". At the bottom, there is a section for "Recent changes" and "Edit this page".

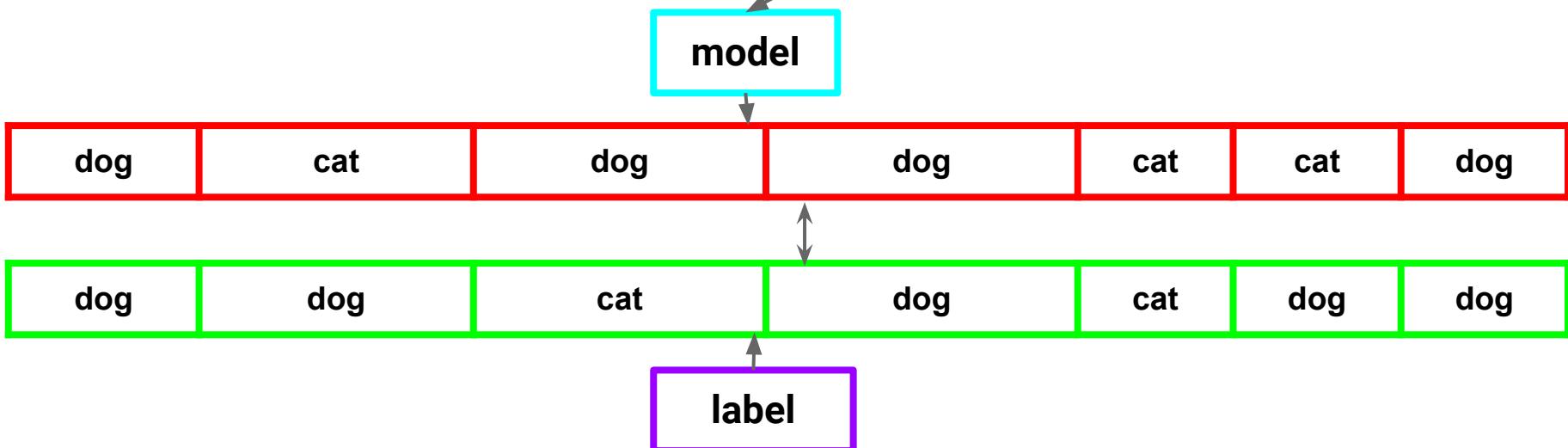
Multiclass classification *(more than one thing or another)*

(multiple label options per sample)

Multilabel classification

Cách đánh giá mô hình trong Supervised Learning

Classification: dog vs cat



Classification metrics

dog	cat	dog	dog	cat	cat	dog
dog	dog	cat	dog	cat	dog	dog

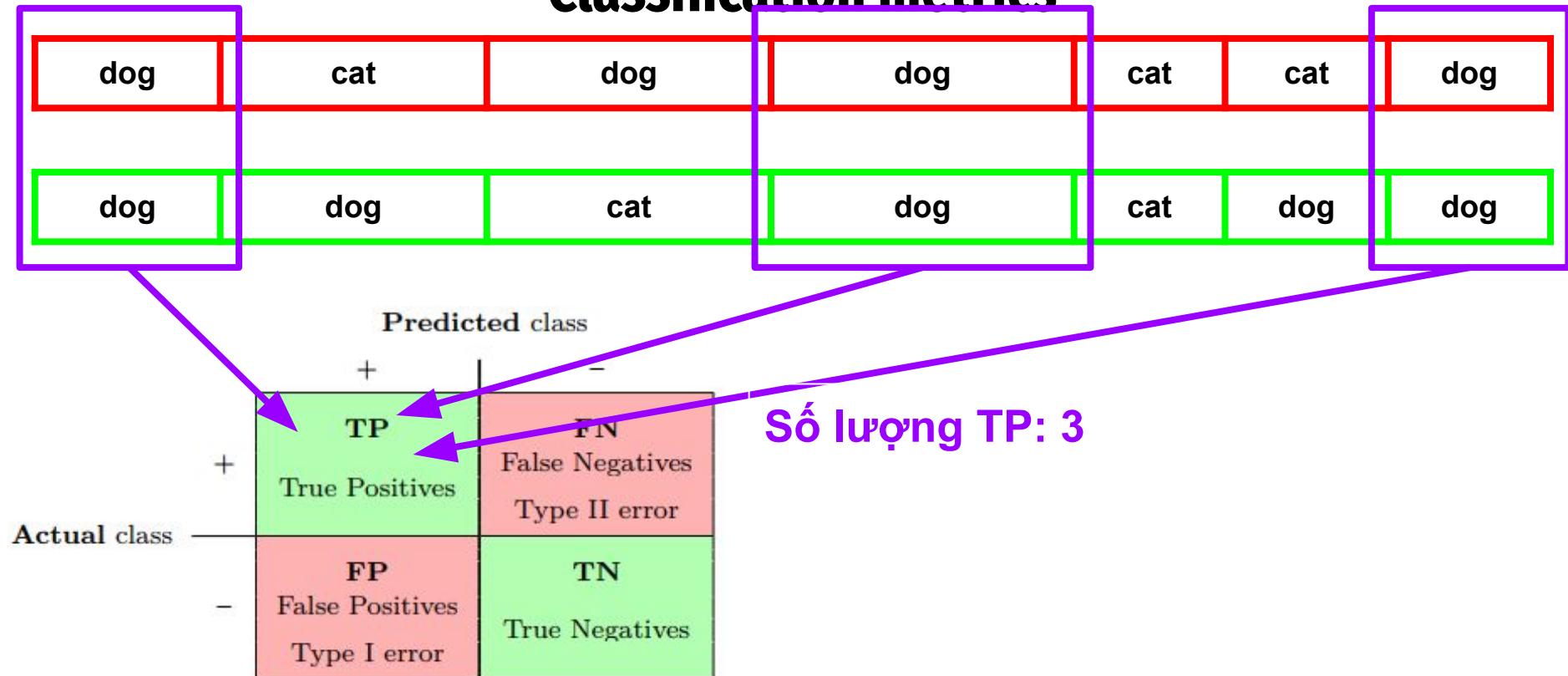
Predicted class

		+	-
Actual class	+	TP True Positives	FN False Negatives Type II error
	-	FP False Positives Type I error	TN True Negatives

Class +: Class ta quan tâm: e.g. dog

Class -: class(es) còn lại: e.g. cat

Classification metrics



Classification metrics

dog	cat	dog	dog	cat	cat	dog
dog	dog	cat	dog	cat	dog	dog

Predicted class

		+	-		
Actual class	+	TP True Positives	FN False Negatives Type II error		
	-	FP False Positives Type I error	TN True Negatives		

Số lượng TP: 3
Số lượng FP: 1

Classification metrics

		Predicted class			
		+	-		
Actual class	+	TP True Positives		FN False Negatives Type II error	
	-	FP False Positives Type I error		TN True Negatives	

Số lượng TP: 3

Số lượng FP: 1

Số lượng FN: 2

Classification metrics

dog	cat	dog	dog	cat	cat	cat	dog
dog	dog	cat	dog	cat	dog	dog	dog

Predicted class

		Predicted class	
		+	-
Actual class	+	TP True Positives	FN False Negatives Type II error
	-	FP False Positives Type I error	TN True Negatives

Số lượng TP: 3

Số lượng FP: 1

Số lượng FN: 2

Số lượng TN: 1

Metric	Công thức	Ý nghĩa
Accuracy	$(TP+TN)/All$	Độ chính xác tổng quát

		Predicted class	
		+	-
		TP True Positives	FN False Negatives Type II error
Actual class	+	FP False Positives Type I error	TN True Negatives
	-		

Số lượng TP: 3 **Accuracy = $(3+1)/7$**
Số lượng FP: 1
Số lượng FN: 2
Số lượng TN: 1
All: 7

Metric	Công thức	Ý nghĩa
Accuracy	$(TP+TN)/All$	Độ chính xác tổng quát
Precision	$TP/(TP+FP)$	Độ chính xác đối với class +

		Predicted class	
		+	-
		TP True Positives	FN False Negatives Type II error
Actual class	+	FP False Positives Type I error	TN True Negatives
	-		

Số lượng TP: 3 Accuracy = $(3+1)/7$

Số lượng FP: 1 Precision = $3/(3+1)$

Số lượng FN: 2

Số lượng TN: 1

All: 7

Metric	Công thức	Ý nghĩa
Accuracy	$(TP+TN)/All$	Độ chính xác tổng quát
Precision	$TP/(TP+FP)$	Độ chính xác đối với class +
Recall	$TP/(TP+FN)$	Độ bao phủ đối với các dự đoán về class +

		Predicted class	
		+	-
		TP True Positives	FN False Negatives Type II error
Actual class	+	FP False Positives Type I error	TN True Negatives
	-		

Số lượng TP: 3 Accuracy = $(3+1)/7$

Số lượng FP: 1 Precision = $3/(3+1)$

Số lượng FN: 2 Recall = $3/(3+2)$

Số lượng TN: 1

All: 7

Metric	Công thức	Ý nghĩa
Accuracy	$(TP+TN)/All$	Độ chính xác tổng quát
Precision	$TP/(TP+FP)$	Độ chính xác đối với class +
Recall	$TP/(TP+FN)$	Độ bao phủ đối với các dự đoán về class +
F1 score	$(2*Pre*Re)/(Pre+Re)$: Trung bình điều hòa giữa Precision và Recall	

		Predicted class	
		+	-
		+	TP True Positives
Actual class		-	FN False Negatives Type II error
		-	FP False Positives Type I error
		-	TN True Negatives

Số lượng TP: 3 Accuracy = $(3+1)/7$

Số lượng FP: 1 Precision = $3/(3+1)$

Số lượng FN: 2 Recall = $3/(3+2)$

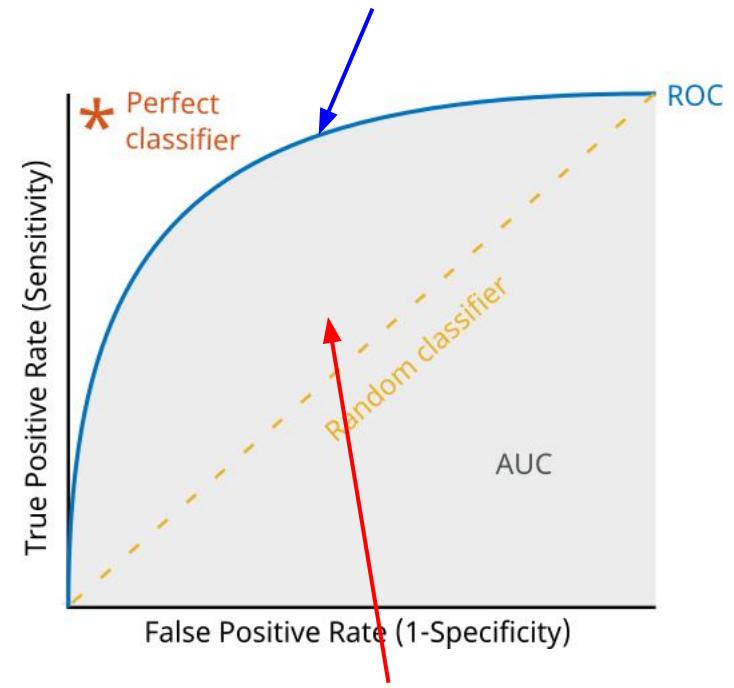
Số lượng TN: 1 F1 score =
All: 7 $(2*0.75*0.6)/(0.75+0.6)$

Classification metrics

		Predicted class	
		+	-
+		TP True Positives	FN False Negatives Type II error
-		FP False Positives Type I error	TN True Negatives

Metric	Công thức
True Positive Rate (= Recall)	$TP/(TP+FN)$
False Positive Rate	$FP/(FP+TN)$

Receiver Operating Characteristic curve

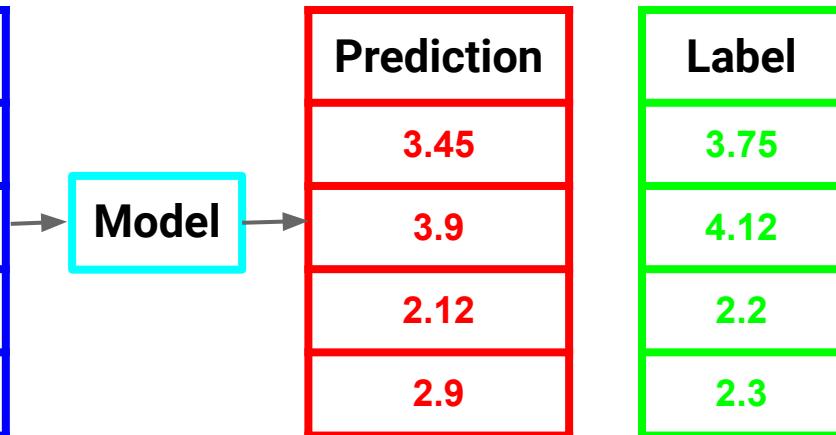


Area under the ROC curve

Cách đánh giá mô hình trong Supervised Learning

Regression: House price prediction

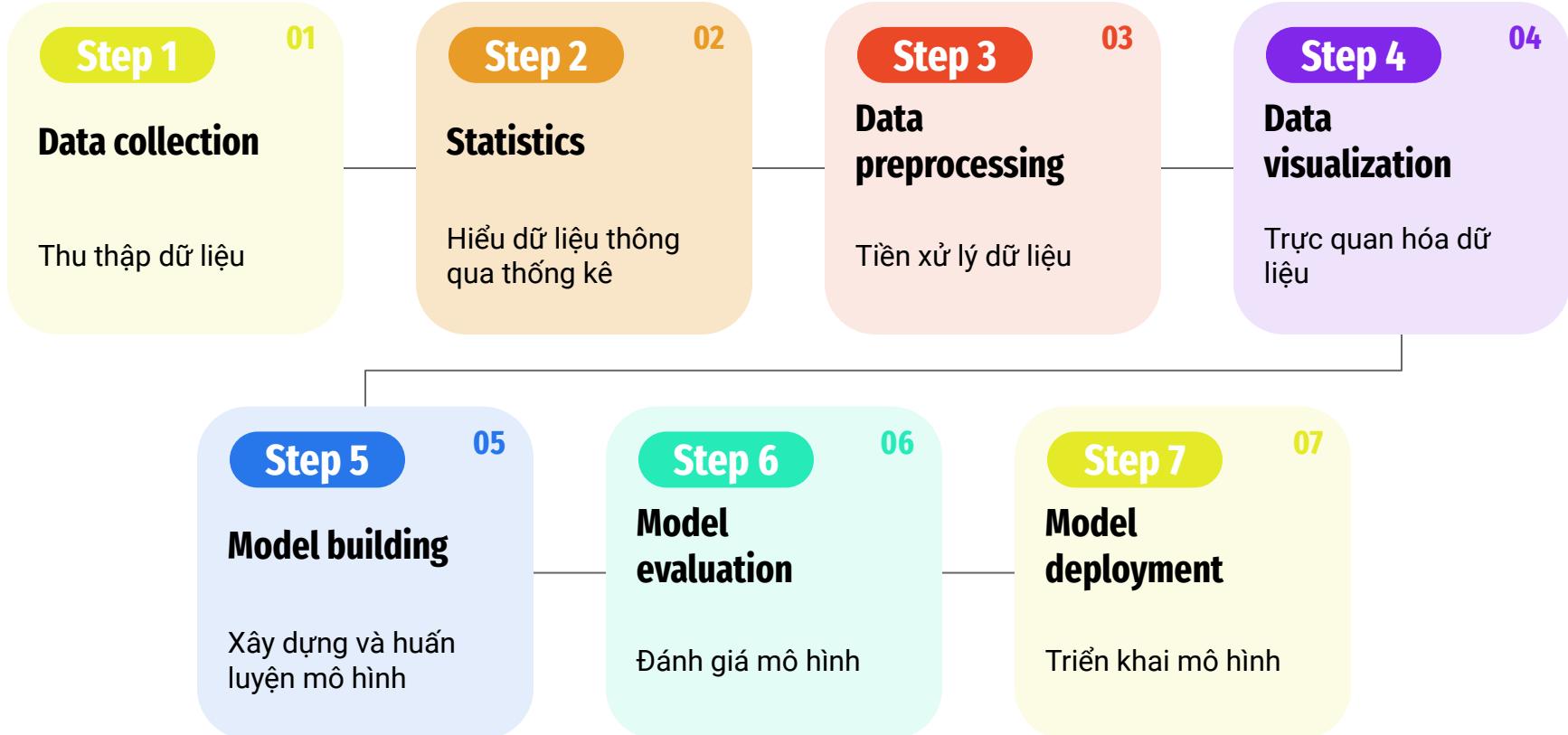
Diện tích	Số tầng	Thành phố	Trong ngõ
40	6	HCM	No
60	4	Hà Nội	No
120	3	Hải Dương	Yes
50	4	Huế	No



$$\text{Mean absolute error} = \frac{1}{N} \sum_{1}^{N} |y_{prediction} - y_{actual}|$$

$$\text{Root Mean Squared Error (RMSE)} = \sqrt{\frac{1}{N} \sum_{1}^{N} (y_{prediction} - y_{actual})^2}$$

Machine Learning pipeline



Step 2: Statistics

Diabetes dataset

Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
6	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
8	183	64	0	0	23.3	0.672	32	1
1	89	66	23	94	28.1	0.167	21	0
0	137	40	35	168	43.1	2.288	33	1
5	116	74	0	0	25.6	0.201	30	0
3	78	50	32	88	31	0.248	26	1
10	115	0	0	0	35.3	0.134	29	0
2	197	70	45	543	30.5	0.158	53	1
8	125	96	0	0	0	0.232	54	1
4	110	92	0	0	37.6	0.191	30	0
10	168	74	0	0	38	0.537	34	1
10	139	80	0	0	27.1	1.441	57	0
1	189	60	23	846	30.1	0.398	59	1
5	166	72	19	175	25.8	0.587	51	1

Step 2: Statistics

How data looks like

The screenshot shows a Jupyter Notebook environment with a Python code cell and a data preview cell.

Code Cell (In [4]):

```
import pandas as pd

data = pd.read_csv("diabetes.csv", delimiter= ",")
```

Data Preview Cell (In [5]):

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6000	0.62700	50	1
1	1	85	66	29	0	26.6000	0.35100	31	0
2	8	183	64	0	0	23.3000	0.67200	32	1
3	1	89	66	23	94	28.1000	0.16700	21	0
4	0	137	40	35	168	43.1000	2.28800	33	1
5	5	116	74	0	0	25.6000	0.20100	30	0
6	3	78	50	32	88	31.0000	0.24800	26	1
7	10	115	0	0	0	35.3000	0.13400	29	0
8	2	197	70	45	543	30.5000	0.15800	53	1
9	8	125	96	0	0	0.0000	0.23200	54	1
10	4	110	92	0	0	37.6000	0.19100	30	0
11	10	168	74	0	0	38.0000	0.53700	34	1
12	10	139	80	0	0	27.1000	1.44100	57	0
13	1	189	60	23	846	30.1000	0.39800	59	1
14	5	166	72	19	175	25.8000	0.58700	51	1
15	7	100	0	0	0	30.0000	0.48400	32	1
16	0	118	84	47	230	45.8000	0.55100	31	1
17	7	107	74	0	0	29.6000	0.25400	31	1
18	1	103	30	38	83	43.3000	0.18300	33	0
19	1	115	70	30	96	34.6000	0.52900	32	1
20	3	126	88	41	235	39.3000	0.70400	27	0
21	8	99	84	0	0	35.4000	0.38800	50	0
22	7	196	90	0	0	38.8000	0.45100	41	1
23	9	119	80	35	0	29.0000	0.26300	29	1
24	11	143	94	33	146	36.6000	0.25400	51	1
25	10	125	70	26	115	31.1000	0.20500	41	1
26	7	147	76	0	0	39.4000	0.25700	43	1
27	1	97	66	15	140	23.2000	0.48700	22	0
28	13	145	82	19	110	22.2000	0.24500	57	0
29	5	117	92	0	0	34.1000	0.33700	38	0
30	5	109	75	26	0	36.0000	0.54600	60	0
31	3	158	76	36	245	31.6000	0.85100	28	1
32	3	88	58	11	54	24.8000	0.26700	22	0
33	6	92	92	0	0	19.9000	0.18800	28	0

Python Console:

```
In [4]: import pandas as pd
...
...: data = pd.read_csv("diabetes.csv", delimiter= ",")
...
...:
```

Structure

Notifications

Format: %%

Colored cells

Resize automatically

Close

Step 2: Statistics

Check each feature's data type

The screenshot shows a Python IDE interface with a menu bar (File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help) and a toolbar. A project named 'MLProject' contains a file 'example.py'. The code in 'example.py' reads a CSV file 'diabetes.csv' and prints the data types of the columns:

```
import pandas as pd

data = pd.read_csv("diabetes.csv", delimiter=",")
print(data.dtypes)
```

The output window shows the data types for each column:

Column	Data Type
Pregnancies	int64
Glucose	int64
BloodPressure	int64
SkinThickness	int64
Insulin	int64
BMI	float64
DiabetesPedigreeFunction	float64
Age	int64
Outcome	int64
dtype: object	

- Tất cả các features và target đều là numerical data
- Có 2 features là float, còn lại là int

Step 2: Statistics

Statistical Summary of Data

The screenshot shows a Python development environment with the following details:

- Project:** MLProject
- File:** example.py
- Code:**

```
from pandas import read_csv

data = read_csv("diabetes.csv")
out = data.describe()
```
- Output Window:** A modal window titled "out" displays the statistical summary of the "diabetes.csv" dataset. The table includes columns: Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age, and Outcome. The "Outcome" column is present but contains no numerical data.

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.00000	768.00000	768.00000	768.00000	768.00000	768.00000	768.00000	768.00000	
mean	3.84505	120.89453	69.10547	20.53649	79.9948	31.99258	0.47188	33.24089	0.34896
std	3.36958	31.97262	19.35581	15.95222	115.24400	7.88416	0.33133	11.76023	0.47695
min	0.00000	0.00000	0.00000	0.00000	0.00000	0.07800	21.00000	0.00000	
25%	1.00000	99.00000	62.00000	0.00000	0.00000	27.30000	0.24975	24.00000	0.00000
50%	3.00000	117.00000	72.00000	23.00000	30.50000	32.00000	0.37250	29.00000	0.00000
75%	6.00000	140.25000	80.00000	32.00000	127.25000	36.60000	0.62625	41.00000	1.00000
max	17.00000	199.00000	122.00000	99.00000	846.00000	67.10000	2.42000	81.00000	1.00000

- Python Console:** Shows the execution of the code:

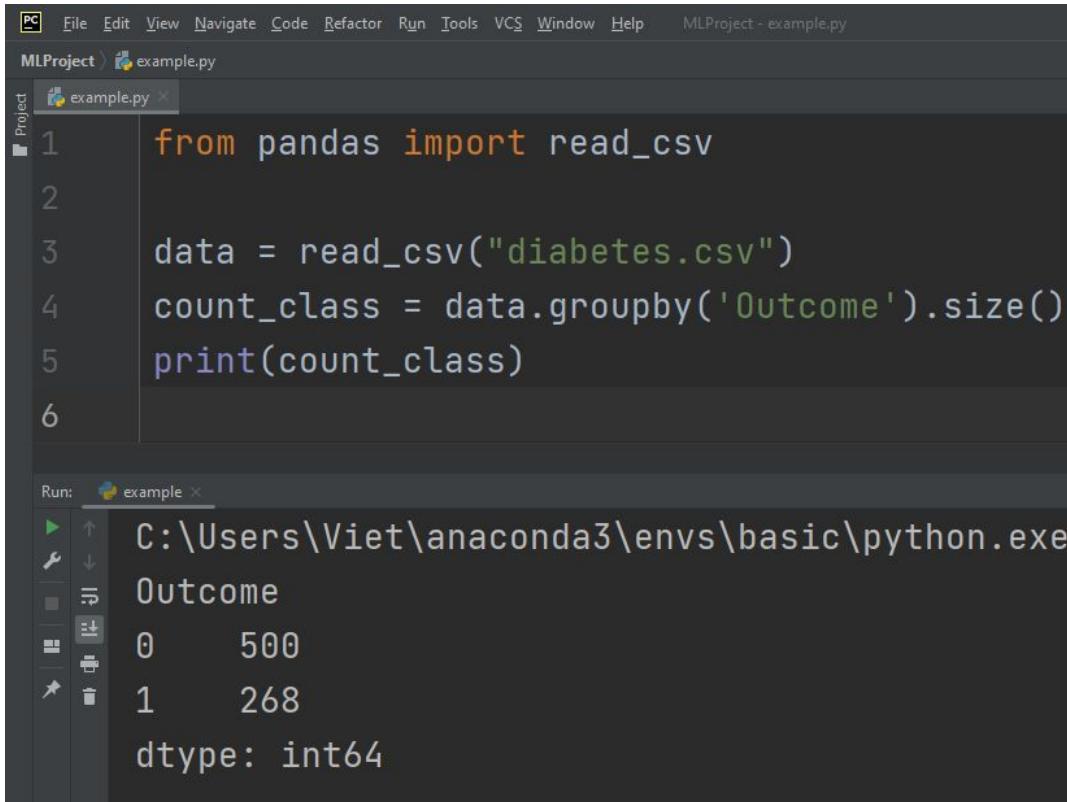
```
In [9]:
```

 followed by the output:

```
Out[9]:
```
- Variable Explorer:** Shows the current state of variables, including:
 - `data` (DataFrame)
 - `out` (DataFrame)
 - `set_option` (CallableDynamicDoc)
 - `Special Variables` (including `_name_`, `_builtins_`, `_doc_`, `_builin_`, `_loader_`, `_spec_`, `_package_`, `_read_csv`, `pd`, and `sys`)

Step 2: Statistics

Category distribution



The screenshot shows a PyCharm IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help, and MLProject - example.py. The left sidebar shows a Project view with a file named example.py. The main code editor window displays the following Python script:

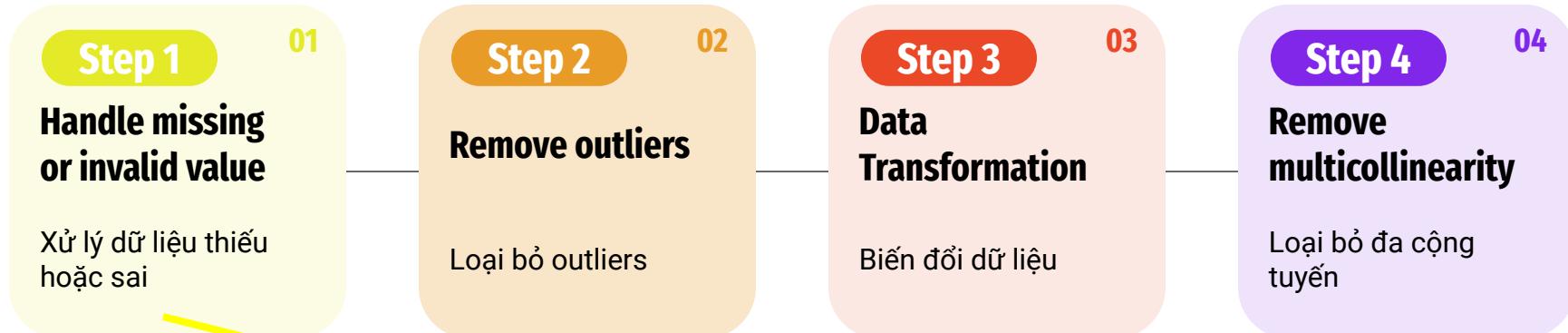
```
from pandas import read_csv  
  
data = read_csv("diabetes.csv")  
count_class = data.groupby('Outcome').size()  
print(count_class)
```

The Run tab at the bottom shows the command: C:\Users\Viet\anaconda3\envs\basic\python.exe. The output pane shows the execution results:

Outcome	Count
0	500
1	268

dtype: int64

Step 3.1: Handle missing or invalid values



Missing values

PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	male	22	1	0	A/5 21171	7.25		S
2	1	1	female	38	1	0	PC 17599	71.2033	C85	C
3	1	3	female	26	0	0	STON/O2. 3101282	7.925		S
4	1	1	female	35	1	0	113803	53.1	C123	S
5	0	3	male	35	0	0	373450	8.05		S
6	0	3	male		0	0	330877	8.4583		Q

Step 3.2: Remove outliers

Step 1

01

**Handle missing
or invalid value**

Xử lý dữ liệu thiếu
hoặc sai

Step 2

02

Remove outliers

Loại bỏ outliers

Step 3

03

**Data
Transformation**

Biến đổi dữ liệu

Step 4

04

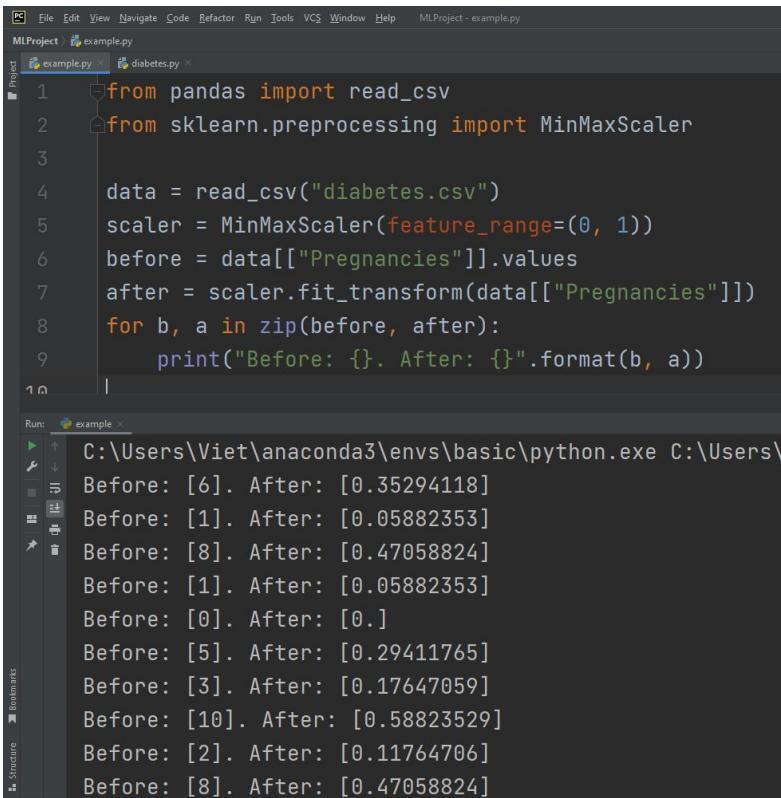
**Remove
multicollinearity**

Loại bỏ đa cộng
tuyến



Step 3.3: Tiền xử lý numerical features

MinMaxScaler



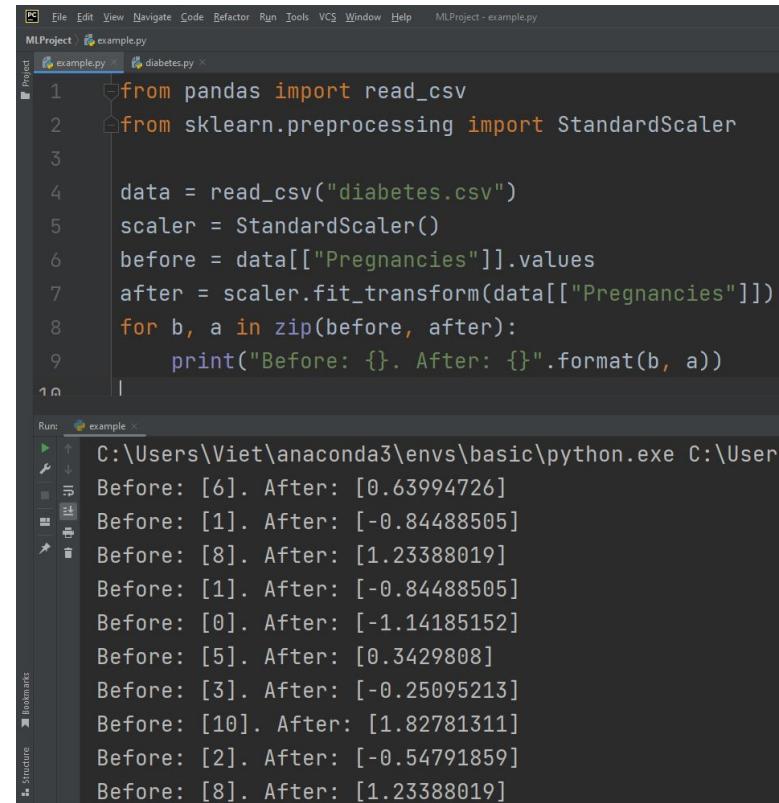
A screenshot of the PyCharm IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help, and MLProject - example.py. The left sidebar shows a project structure with files example.py and diabetes.py. The main code editor contains Python code demonstrating the use of MinMaxScaler on the 'Pregnancies' feature of the diabetes dataset. The run output at the bottom shows the transformation of various values from their original range to a new range between 0 and 1.

```
from pandas import read_csv
from sklearn.preprocessing import MinMaxScaler

data = read_csv("diabetes.csv")
scaler = MinMaxScaler(feature_range=(0, 1))
before = data[["Pregnancies"]].values
after = scaler.fit_transform(data[["Pregnancies"]])
for b, a in zip(before, after):
    print("Before: {}. After: {}".format(b, a))
```

Run: example x
C:\Users\Viet\anaconda3\envs\basic\python.exe C:\Users\Viet\MLProject\example.py
Before: [6]. After: [0.35294118]
Before: [1]. After: [0.05882353]
Before: [8]. After: [0.47058824]
Before: [1]. After: [0.05882353]
Before: [0]. After: [0.]
Before: [5]. After: [0.29411765]
Before: [3]. After: [0.17647059]
Before: [10]. After: [0.58823529]
Before: [2]. After: [0.11764706]
Before: [8]. After: [0.47058824]

StandardScaler



A screenshot of the PyCharm IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help, and MLProject - example.py. The left sidebar shows a project structure with files example.py and diabetes.py. The main code editor contains Python code demonstrating the use of StandardScaler on the 'Pregnancies' feature of the diabetes dataset. The run output at the bottom shows the transformation of various values from their original range to a new range centered around 0 with a standard deviation of 1.

```
from pandas import read_csv
from sklearn.preprocessing import StandardScaler

data = read_csv("diabetes.csv")
scaler = StandardScaler()
before = data[["Pregnancies"]].values
after = scaler.fit_transform(data[["Pregnancies"]])
for b, a in zip(before, after):
    print("Before: {}. After: {}".format(b, a))
```

Run: example x
C:\Users\Viet\anaconda3\envs\basic\python.exe C:\Users\Viet\MLProject\example.py
Before: [6]. After: [0.63994726]
Before: [1]. After: [-0.84488505]
Before: [8]. After: [1.23388019]
Before: [1]. After: [-0.84488505]
Before: [0]. After: [-1.14185152]
Before: [5]. After: [0.3429808]
Before: [3]. After: [-0.25095213]
Before: [10]. After: [1.82781311]
Before: [2]. After: [-0.54791859]
Before: [8]. After: [1.23388019]

Step 3.3: Tiền xử lý ordinal features

OrdinalEncoder

The screenshot shows the PyCharm IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help, and MLProject - example.py. The left sidebar shows a Project view with 'example.py' selected. The main code editor contains the following Python script:

```
from pandas import DataFrame
from sklearn.preprocessing import OrdinalEncoder

before = DataFrame(["XL", "L", "S", "M", "XS", "XS", "L", "M"])
values = ["XS", "S", "M", "L", "XL"]
scaler = OrdinalEncoder(categories=[values])
after = scaler.fit_transform(before)
for b, a in zip(before.values, after):
    print("Before: {}. After: {}".format(b, a))
```

The 'Run' tool window at the bottom shows the command: C:\Users\Viet\anaconda3\envs\basic\python.exe C:\Users\Viet\Pycha... and the output of the script:

```
Before: ['XL']. After: [4.]
Before: ['L']. After: [3.]
Before: ['S']. After: [1.]
Before: ['M']. After: [2.]
Before: ['XS']. After: [0.]
Before: ['XS']. After: [0.]
Before: ['L']. After: [3.]
Before: ['M']. After: [2.]
```

Step 3.3: Các cách mã hóa nominal features

One-hot encoding

Color

- Red -> 1, 0, 0, 0
- Green -> 0, 1, 0, 0
- Blue -> 0, 0, 1, 0
- Black -> 0, 0, 0, 1

Advantage

- Simple

Disadvantage

- Sparse data

Hash encoding

Country

- USA -> 1, 0, 0
- UK -> 0, 0, 1
- Korea -> 0, 1, 0
- Russia -> 1, 0, 0

Advantage

- Simple
- Less sparse

Disadvantage

- Collision
- No inverse-mapping

Word Embedding

Animal

- Dog -> 0.27, -0.31, -0.53
- Lion -> -0.7, 0.61, 0.42
- Tiger -> -0.71, 0.6, 0.38
- Mouse -> 0.31, -0.34, 0.76

Advantage

- Memory efficient
- Relationship learnt

Disadvantage

- Word2vec model needed
- Maybe many dimensions needed

Step 3.4: Mối quan hệ giữa các variables (features, target)

Correlation

- Mức độ 2 hay nhiều variables quan hệ tuyến tính với nhau
- Hệ số tương quan: [-1, 1]
- Correlation does not imply causation

$$r_{xy} = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}}$$

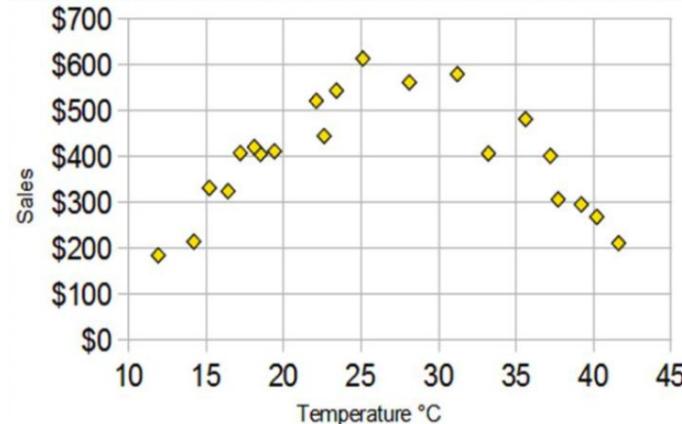
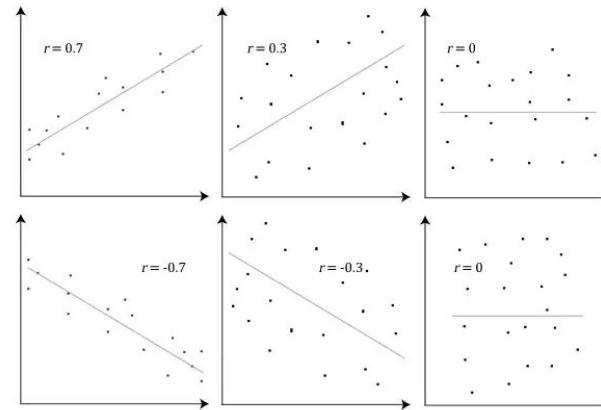
r_{xy} = correlation coefficient between x and y

x_i = the values of x within a sample

y_i = the values of y within a sample

\bar{x} = the average of the values of x within a sample

\bar{y} = the average of the values of y within a sample



Step 3.4: Mối quan hệ giữa các features

(Multi)collinearity

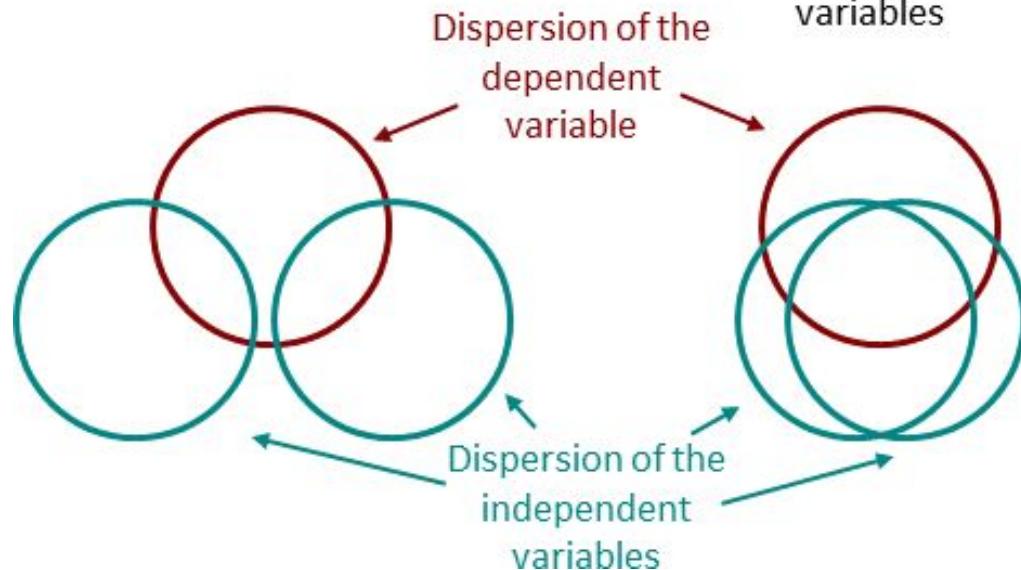
- Mức độ 2 hay nhiều **features** quan hệ tuyến tính với nhau
- Thông thường hệ số tương quan > 0.7 hoặc < -0.7 biểu thị rằng 2 hay nhiều features có hiện tượng (đa) cộng tuyến với nhau

No multicollinearity

No correlation of the independent variables

Multicollinearity

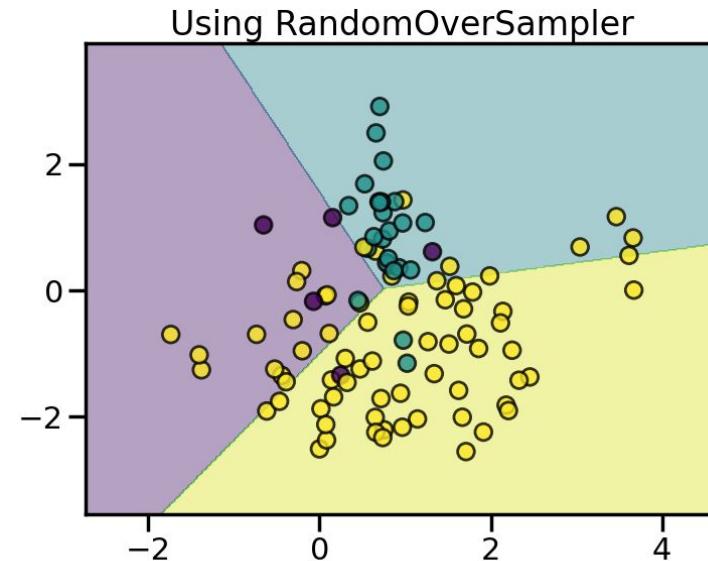
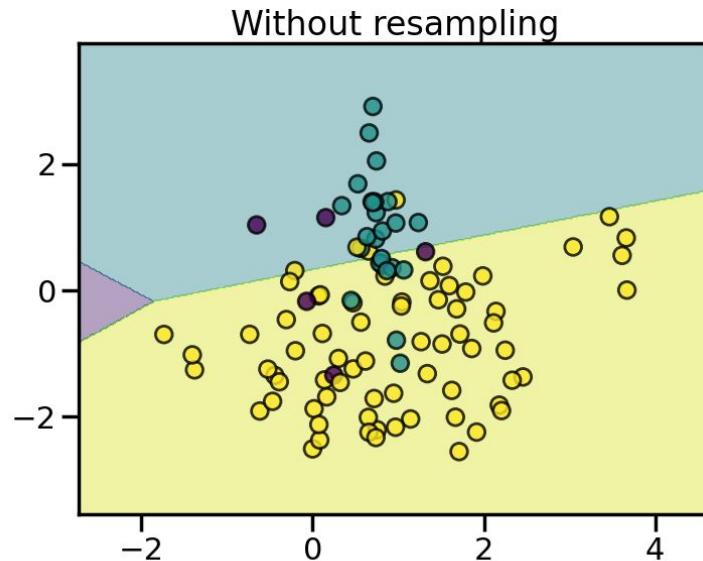
High correlation of the independent variables



Step 3.5: Balance data

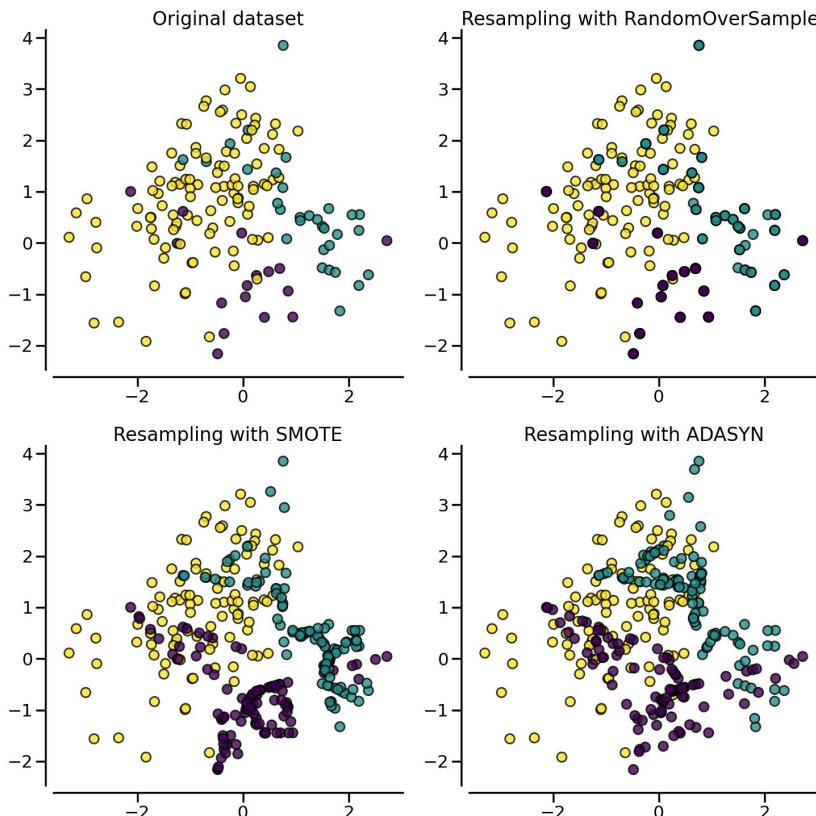
Over-sampling

Decision function of LogisticRegression



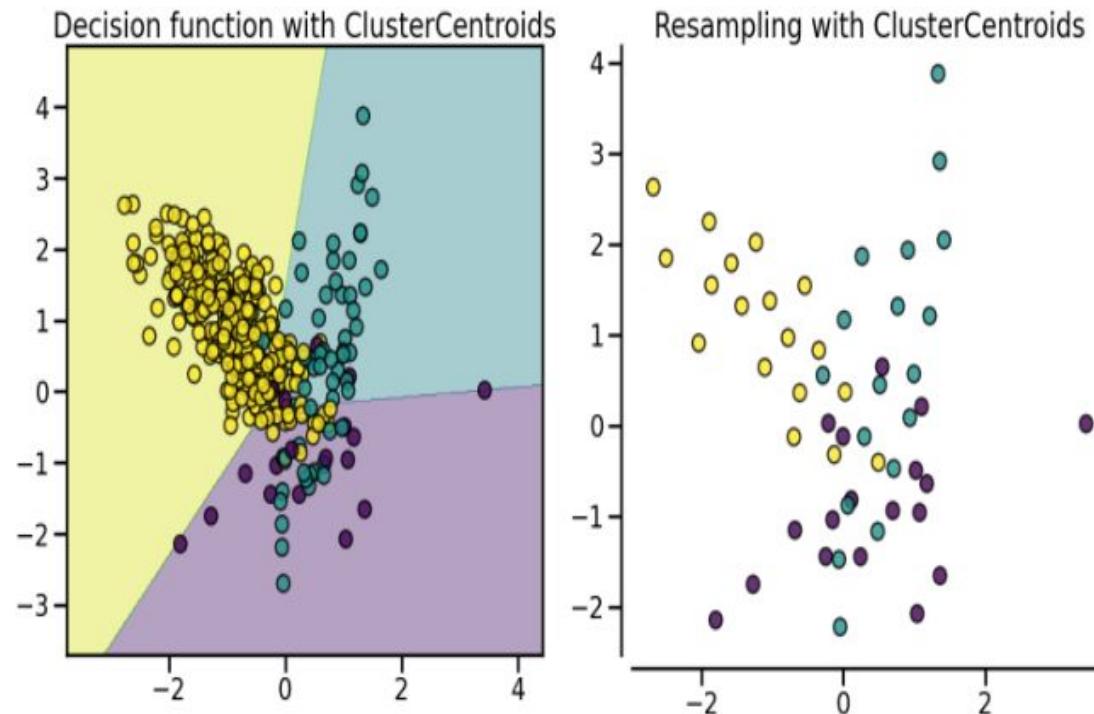
Step 3.5: Balance data

Over-sampling



Step 3.5: Balance data

Under-sampling



Step 4: Data visualization



matplotlib



seaborn

Histogram

The screenshot shows a Python development environment with a code editor and a terminal window. The code in the editor is as follows:

```
from pandas import read_csv
import matplotlib.pyplot as plt

data = read_csv("diabetes.csv")
data.hist()
plt.show()
```

The terminal window shows the command run: `example` and the path `C:\Users\Viet\anaconda3\envs\basic\python.exe`.

A figure titled "Figure 1" displays a 3x3 grid of histograms for the "diabetes.csv" dataset. The columns represent different features: Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age, and Outcome.

- Pregnancies:** The distribution is right-skewed, with the highest frequency occurring at 0 pregnancies.
- Glucose:** The distribution is unimodal and slightly right-skewed, peaking around 100-125.
- BloodPressure:** The distribution is bimodal, with peaks around 30 and 60.
- SkinThickness:** The distribution is right-skewed, with the highest frequency at 0.
- Insulin:** The distribution is highly right-skewed, with the highest frequency at 0.
- BMI:** The distribution is right-skewed, with the highest frequency around 30.
- DiabetesPedigreeFunction:** The distribution is right-skewed, with the highest frequency at 0.
- Age:** The distribution is highly right-skewed, with the highest frequency at 20.
- Outcome:** The distribution is bimodal, with peaks at 0.00 and 1.00.

Density Plot

File Edit View Navigate Code Refactor Run Tools VCS Window Help MLProject - example.py

MLProject example.py

```
1 from pandas import read_csv
2
3
4 data = read_csv("diabetes.csv")
5 data.plot(kind='density', subplots=True, layout=(3, 3), sharex=False)
6 plt.show()
7
```

Figure 1

The figure consists of nine subplots arranged in a 3x3 grid. Each subplot shows the density distribution of a specific feature. The features and their corresponding color-coded distributions are:

- Pregnancies (Blue): A single sharp peak centered around 0.
- Glucose (Orange): A single sharp peak centered around 100.
- BloodPressure (Green): A single sharp peak centered around 50.
- SkinThickness (Red): Two distinct peaks, one at approximately 0 and another at approximately 20.
- Insulin (Purple): A single sharp peak centered around 200.
- BMI (Brown): A single sharp peak centered around 30.
- DiabetesPedigreeFunction (Pink): A single sharp peak centered around 0.5.
- Age (Grey): A single sharp peak centered around 35.
- Outcome (Yellow-Green): Two distinct peaks, one at approximately 0.2 and another at approximately 1.0.

Run: example x C:\Users\Viet\anaconda3\env

Bookmark Structure

Box Plot

The screenshot shows a code editor interface with a Python script named `example.py` open. The script imports `pandas` and `matplotlib.pyplot`, reads a CSV file named `diabetes.csv`, and creates a 3x3 grid of box plots using `data.plot(kind='box', subplots=True, layout=(3, 3), sharex=False)`. The plots are displayed in a separate window titled "Figure 1".

```
from pandas import read_csv
import matplotlib.pyplot as plt

data = read_csv("diabetes.csv")
data.plot(kind='box', subplots=True, layout=(3, 3), sharex=False)
plt.show()
```

Run: example
C:\Users\Viet\anaconda3\envs\basic\python

Project: MLProject / example.py / frame.py

File Edit View Navigate Code Refactor Run Tools VCS Window Help MLProject - example.py

Notifications

Figure 1

<img alt="A 3x3 grid of box plots for the 'diabetes.csv' dataset. The columns represent 'Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', and 'Outcome'. The y-axis scales vary by plot: Pregnancies (0-15), Glucose (0-200), BloodPressure (0-100), SkinThickness (0-100), Insulin (0-800), BMI (0-1.00), DiabetesPedigreeFunction (0-2), Age (0-80), and Outcome (0.00-1.00). Each plot shows the distribution of values for that specific feature, including the median (green line), quartiles (blue boxes), and range (whiskers). Outliers are represented by black dots and circles.</p>

Correlation Matrix Plot

The screenshot shows a Python IDE interface with a project named "MLProject" containing files "example.py" and "frame.py". The code in "example.py" reads a CSV file "diabetes.csv" and calculates its correlation matrix.

```
from pandas import read_csv

data = read_csv("diabetes.csv")
correlations = data.corr()
```

A modal window titled "correlations" displays the correlation matrix. The matrix is a 10x10 grid where each cell's color represents the strength and sign of the correlation between the corresponding columns. Darker colors indicate stronger correlations, while lighter colors indicate weaker ones. The diagonal elements are all 1.00000, representing perfect correlation with themselves.

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1.00000	0.12946	0.14128	-0.08167	-0.07353	0.01768	-0.03352	0.54434	0.22190
Glucose	0.12946	1.00000	0.15259	0.05733	0.33136	0.22107	0.13734	0.26351	0.46658
BloodPressure	0.14128	0.15259	1.00000	0.20737	0.08893	0.28181	0.04126	0.23953	0.06507
SkinThickness	-0.08167	0.05733	0.20737	1.00000	0.43678	0.39257	0.18393	-0.11397	0.07475
Insulin	-0.07353	0.33136	0.08893	0.43678	1.00000	0.19786	0.18507	-0.04216	0.13055
BMI	0.01768	0.22107	0.28181	0.39257	0.19786	1.00000	0.14065	0.03624	0.29269
DiabetesPedigreeFunction	-0.03352	0.13734	0.04126	0.18393	0.18507	0.14065	1.00000	0.03356	0.17384
Age	0.54434	0.26351	0.23953	-0.11397	-0.04216	0.03624	0.03356	1.00000	0.23836
Outcome	0.22190	0.46658	0.06507	0.07475	0.13055	0.29269	0.17384	0.23836	1.00000

Below the matrix, there are input fields for "correlations" and "Format: %s", and checkboxes for "Colored cells" and "Resize automatically".

Correlation Matrix Plot

MLProject

File Edit View Navigate Code Refactor Run Tools VCS Window Help

MLProject - example.py

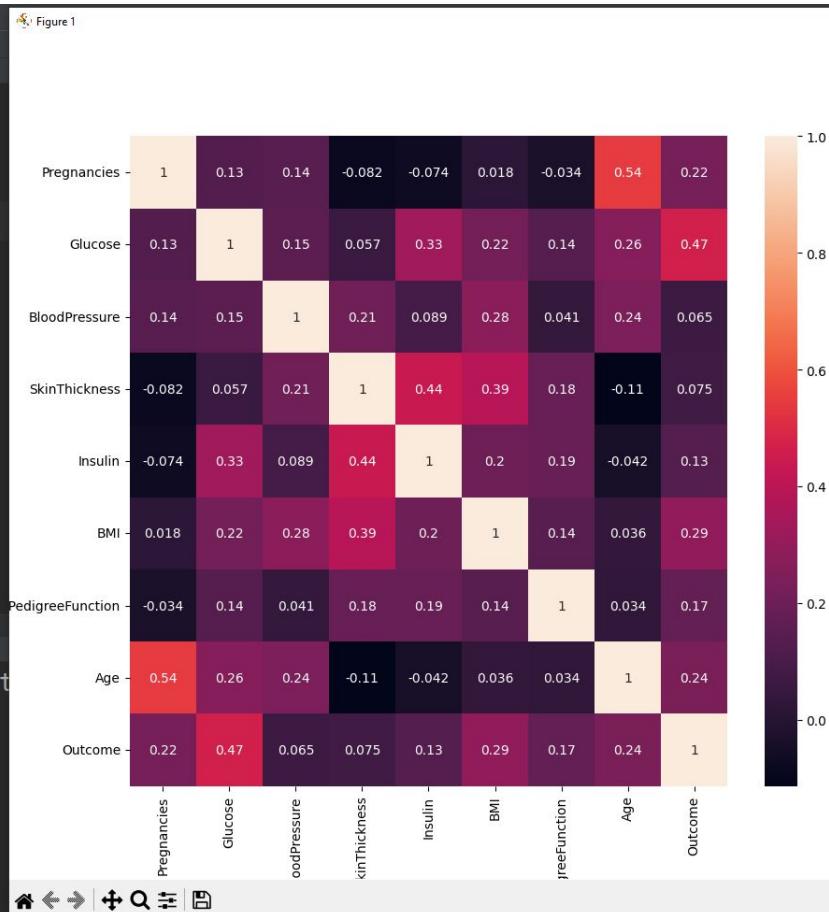
Project

example.py frame.py

```
1 from pandas import read_csv
2
3 import seaborn as sn
4
5 import matplotlib.pyplot as plt
6
7
8 data = read_csv("diabetes.csv")
9 sn.heatmap(data.corr(), annot=True)
10 plt.show()
```

Run: example x

C:\Users\Viet\anaconda3\envs\basic\pyt



Scatter Matrix Plot

File Edit View Navigate Code Refactor Run Tools VCS Window Help MLProject - example1

MLProject example.py

students.py ex2.py example.py

```
1 from pandas import read_csv
2 from pandas.plotting import scatter_matrix
3 import matplotlib.pyplot as plt
4
5 data = read_csv("diabetes.csv")
6 scatter_matrix(data)
7 plt.show()
```

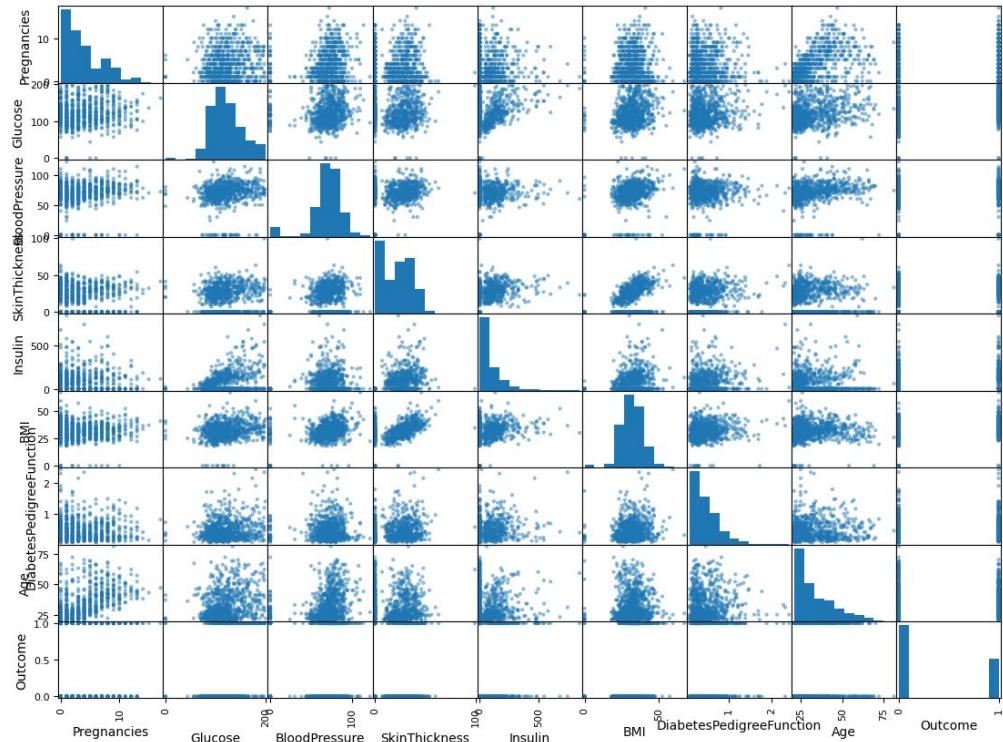
Run: example

C:\Users\Viet\anaconda3\envs\bas...

Structure

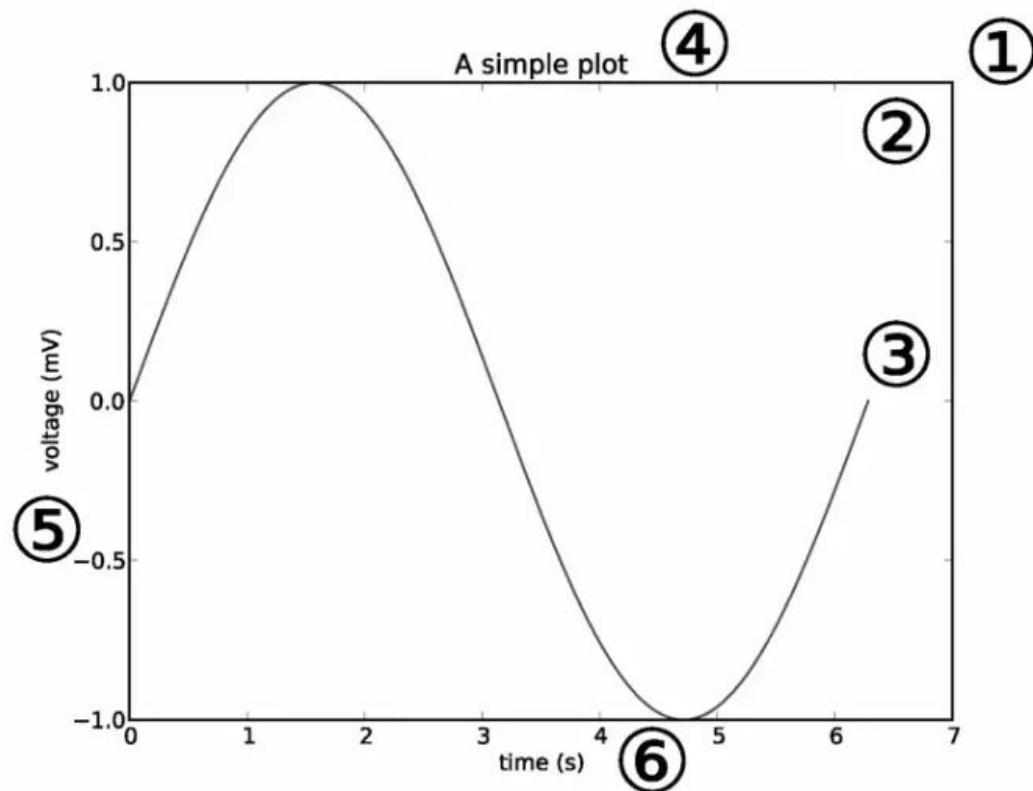
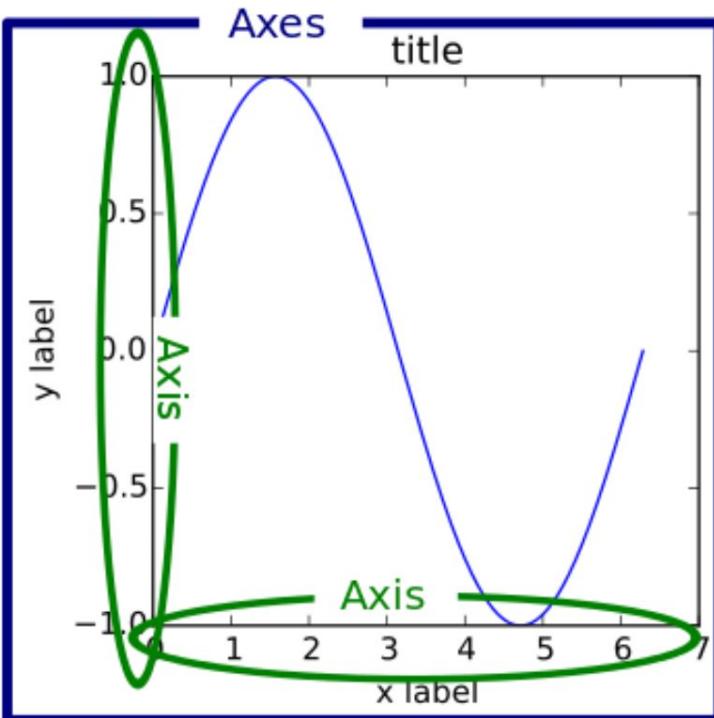
Version Control Run Python Packages TODO Python Console Problems

Download pre-built shared indexes: Reduce the indexing time and CPU load with pre-built Python packages shared

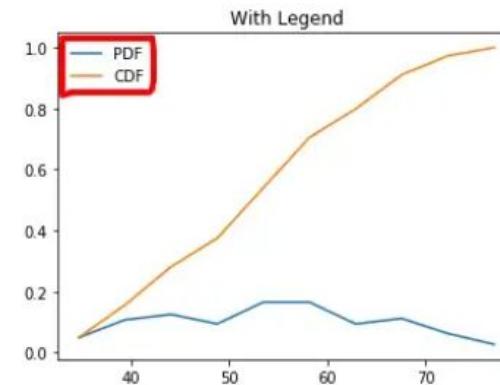
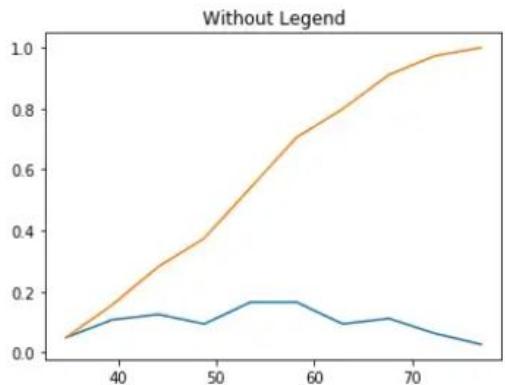
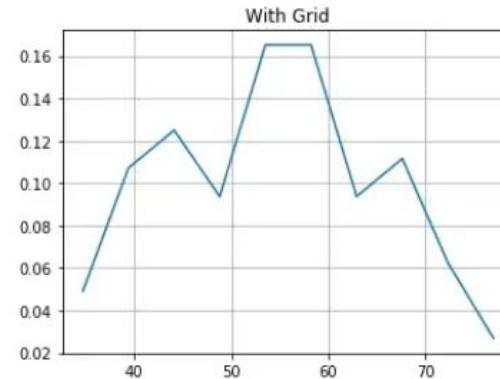
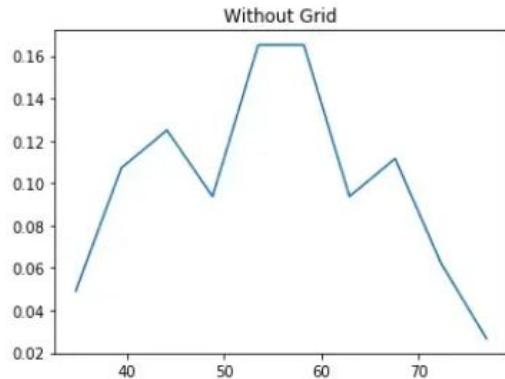


Data visualization with matplotlib

Figure



Data visualization with matplotlib



Supervised Learning algorithms

Regression(hồi quy)

Linear Regression

Polynomial Regression

***Could also be used here**

Classification(Phân loại)

Logistic Regression

Support Vector Machine*

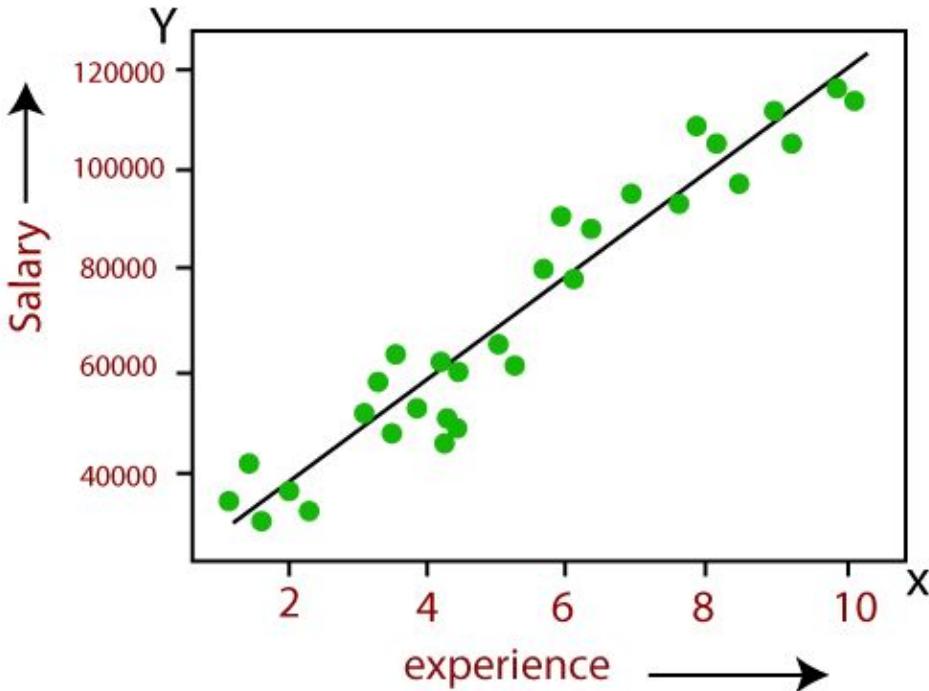
K-nearest neighbours*

Naive Bayes

Decision Tree*

Random Forest*

Linear Regression



- 1 trong số các thuật toán cơ bản và đơn giản nhất trong Supervised Learning
- Simple linear regression: 1 input feature
- Multiple linear regression: 2 input features trở lên
- Phù hợp với các linear dataset

predictor, 'x-variable',
independent variable,
explanatory variable

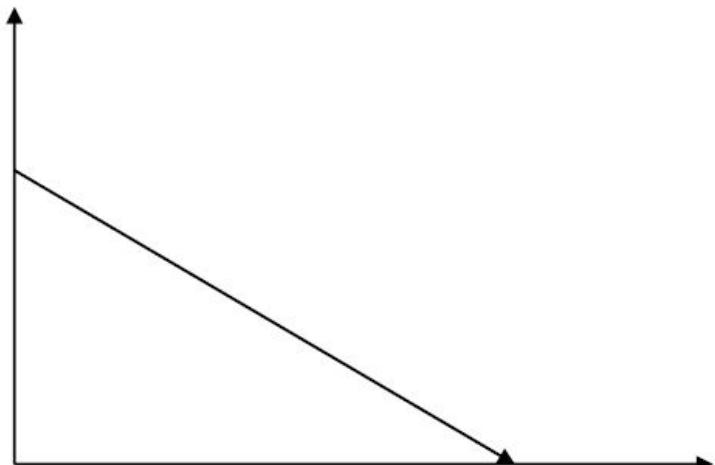
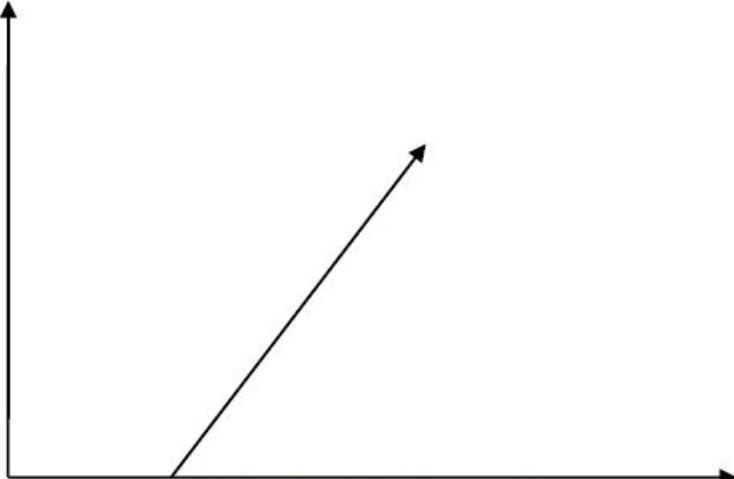
coefficient

$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_p x_p + \varepsilon$$

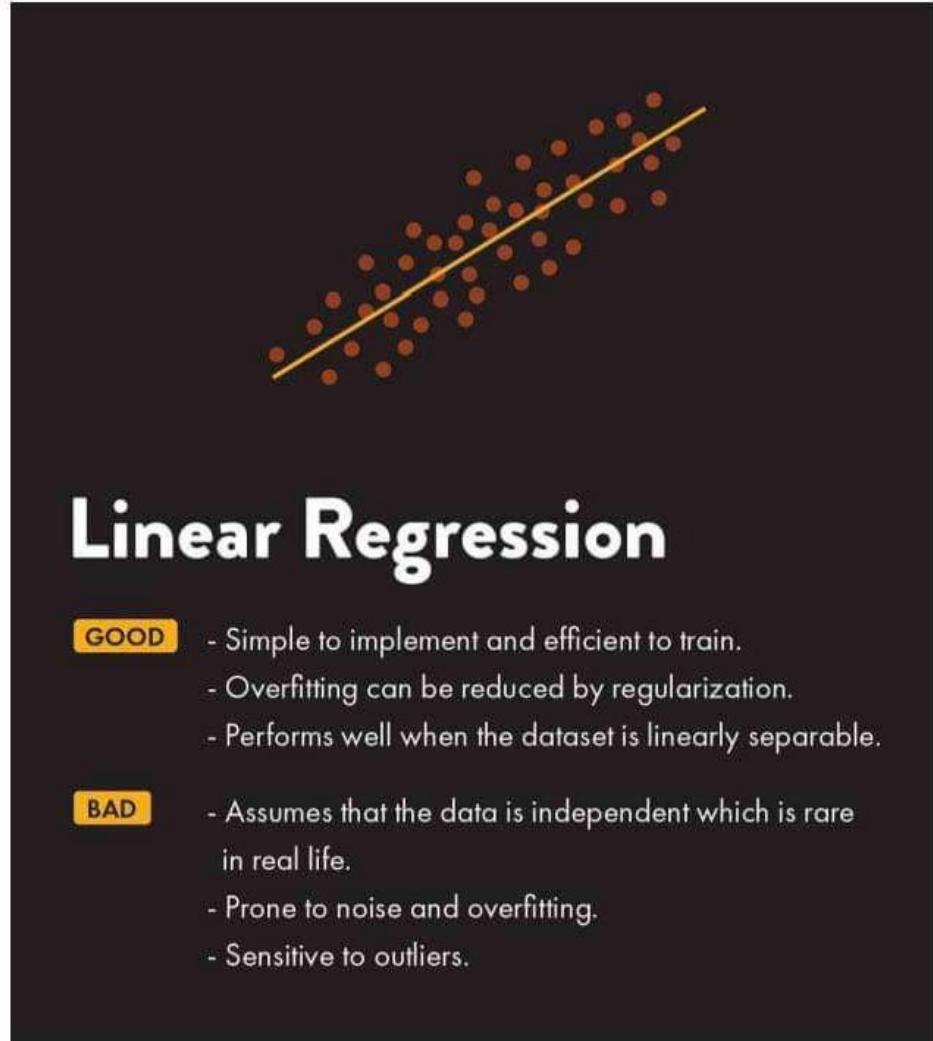
linear predictor

response, dependent variable,
observation, 'y-variable'

random error,
"noise"



Negative Linear Relationship



Linear Regression

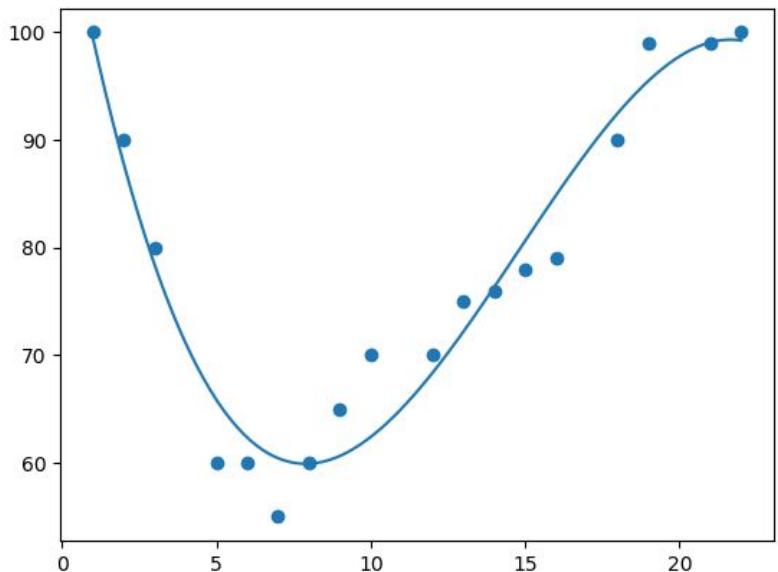
GOOD

- Simple to implement and efficient to train.
- Overfitting can be reduced by regularization.
- Performs well when the dataset is linearly separable.

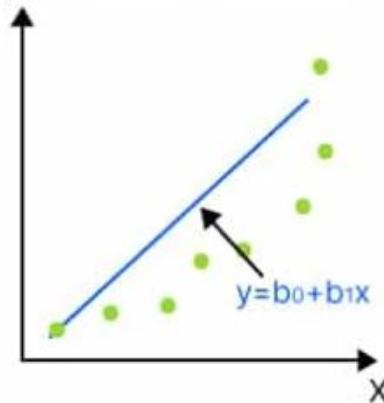
BAD

- Assumes that the data is independent which is rare in real life.
- Prone to noise and overfitting.
- Sensitive to outliers.

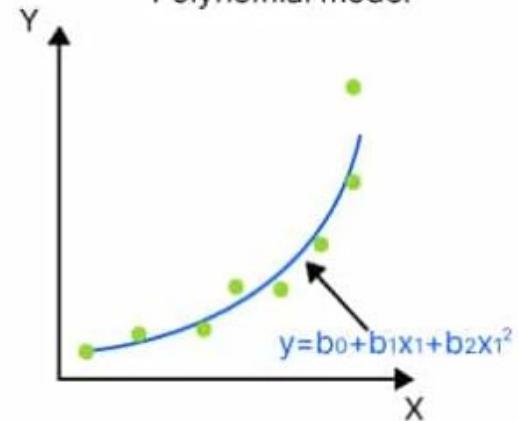
Polynomial Regression



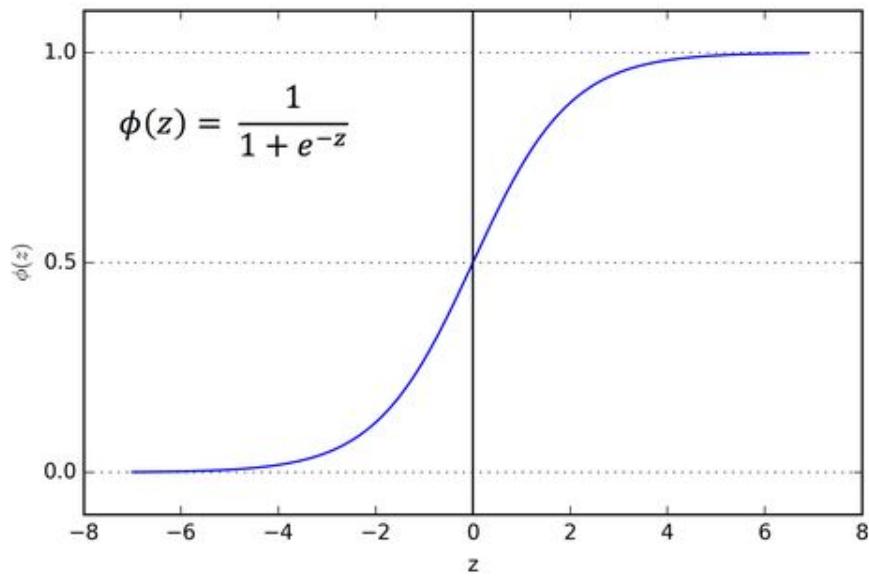
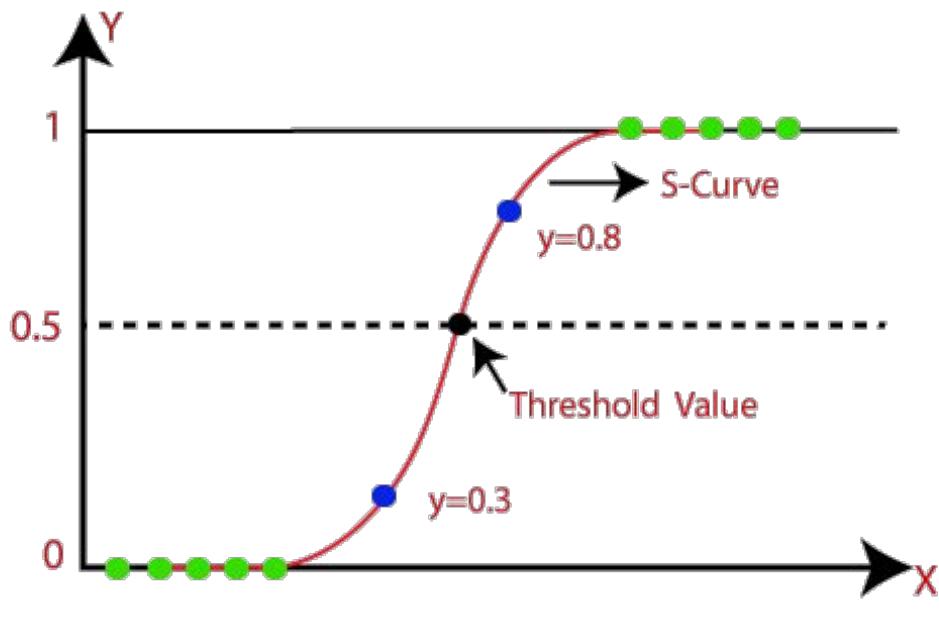
Simple linear model

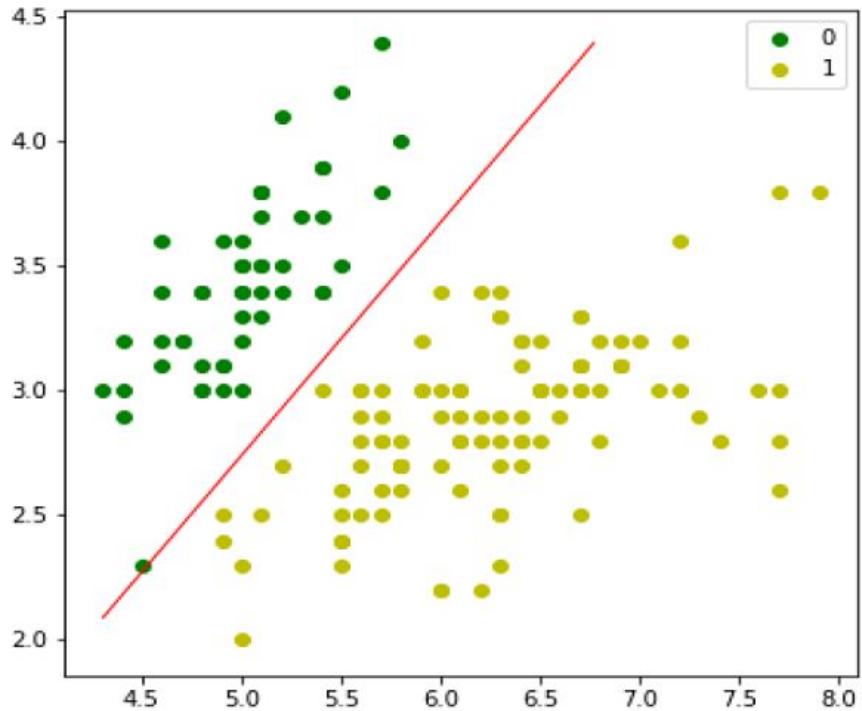


Polynomial model



Logistic Regression





Logistic Regression

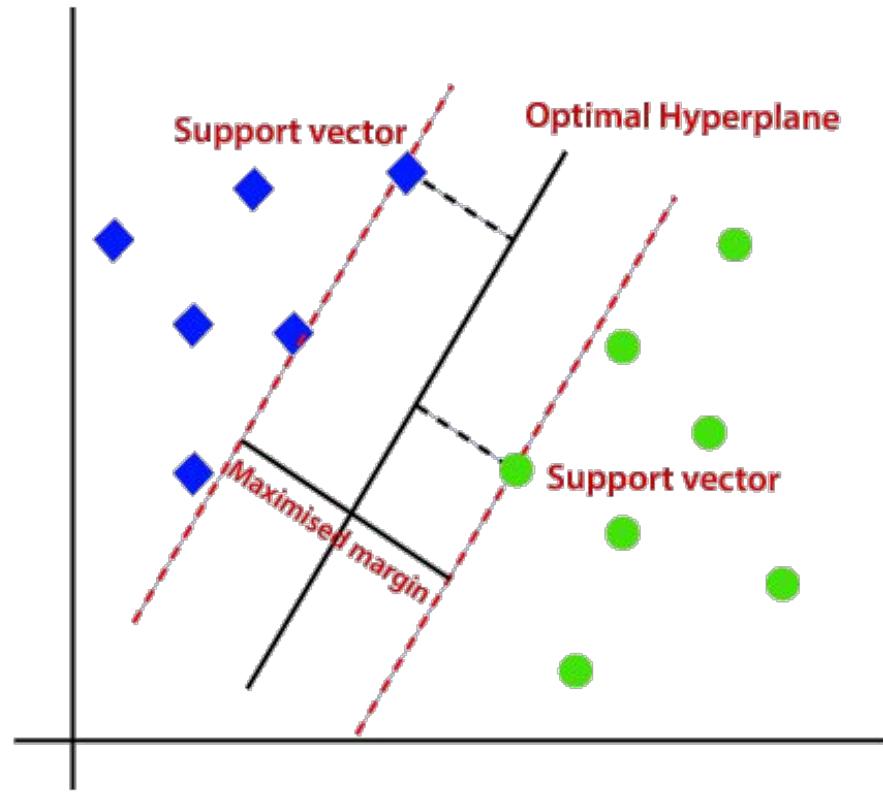
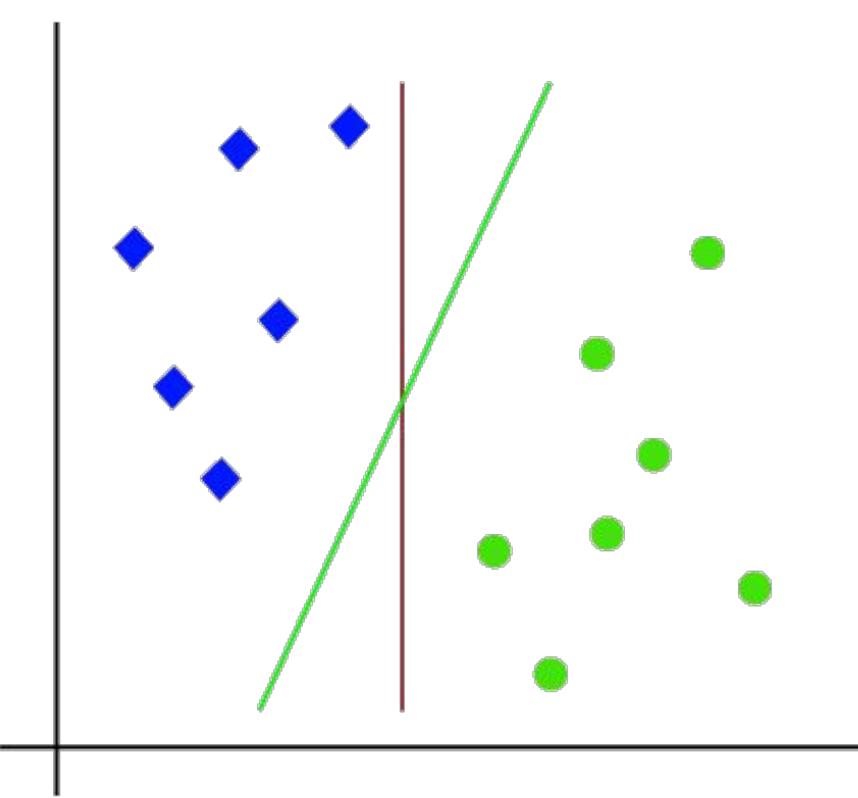
GOOD

- Less prone to over-fitting but it can overfit in high dimensional datasets.
- Efficient when the dataset has features that are linearly separable.
- Easy to implement and efficient to train.

BAD

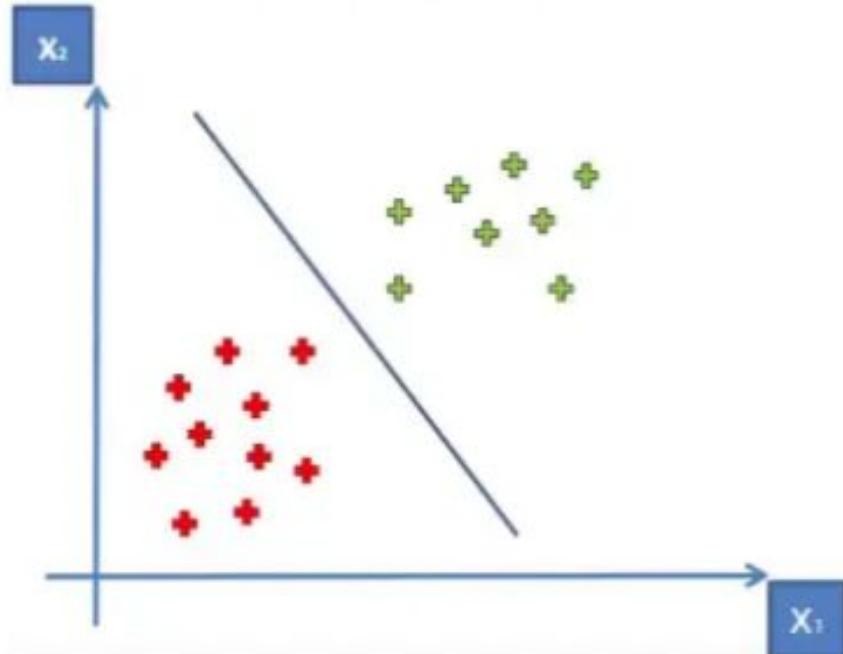
- Should not be used when the number of observations are lesser than the number of features.
- Assumption of linearity which is rare in practise.
- Can only be used to predict discrete functions.

Support Vector Machine

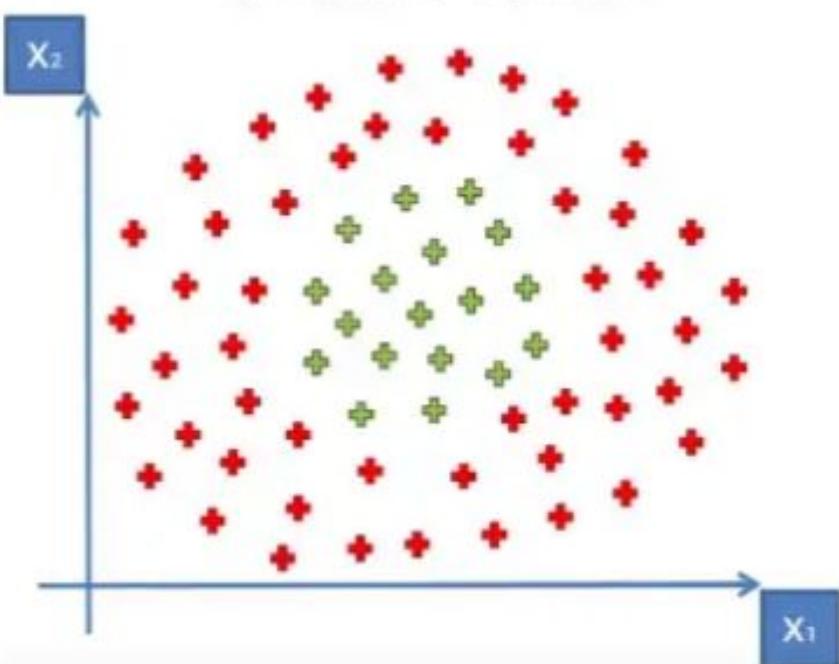


Support Vector Machine

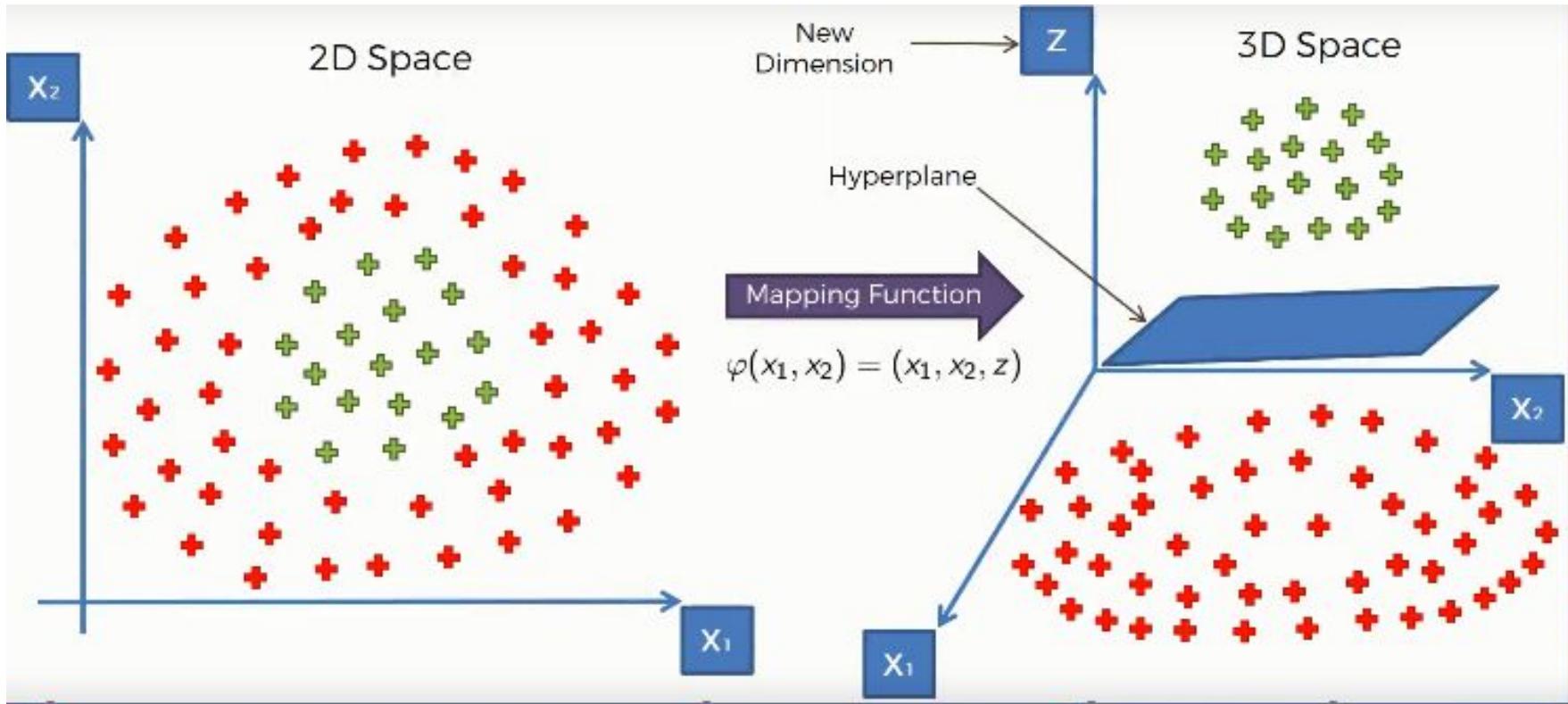
Linearly Separable



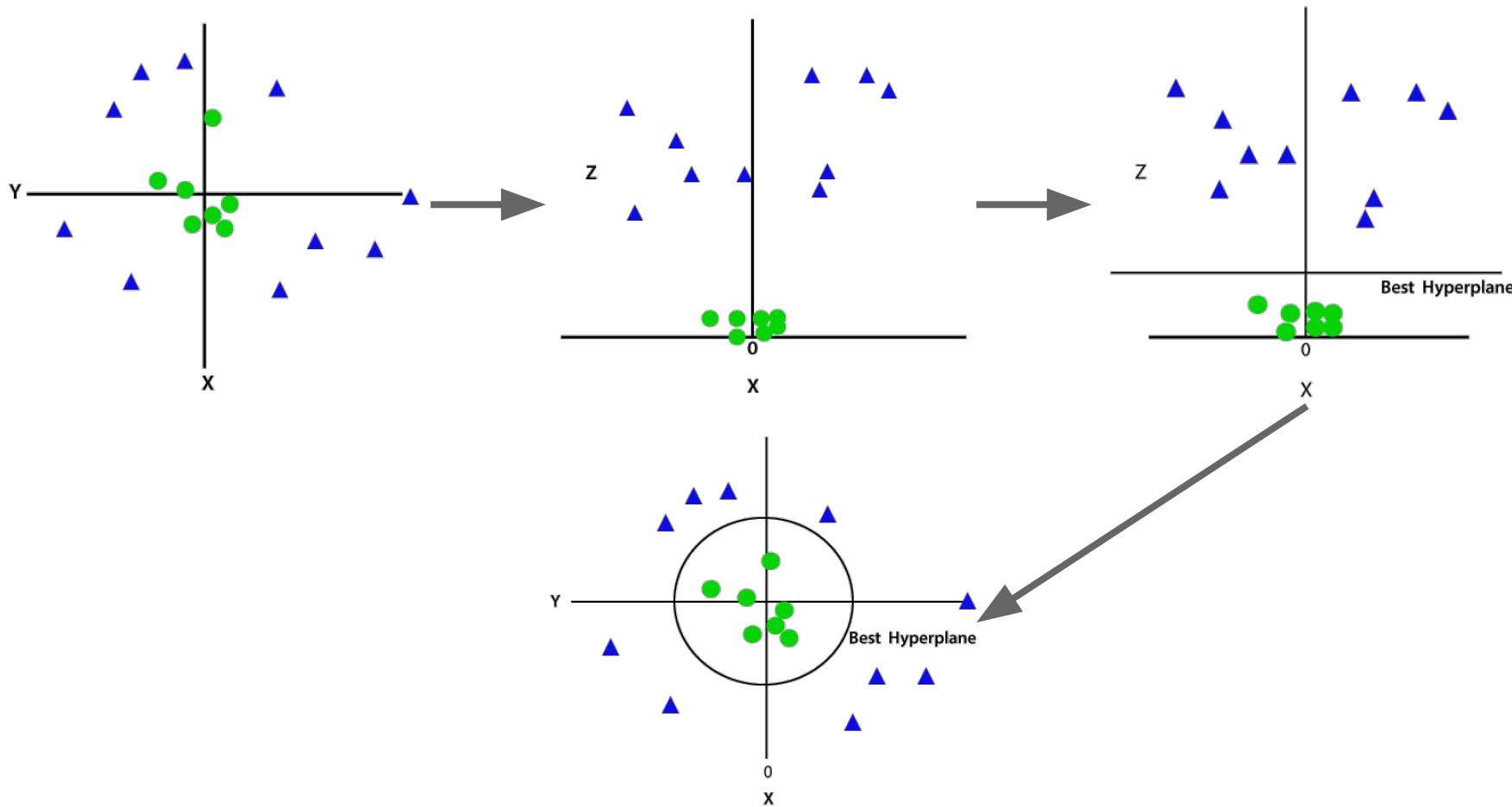
Not Linearly Separable

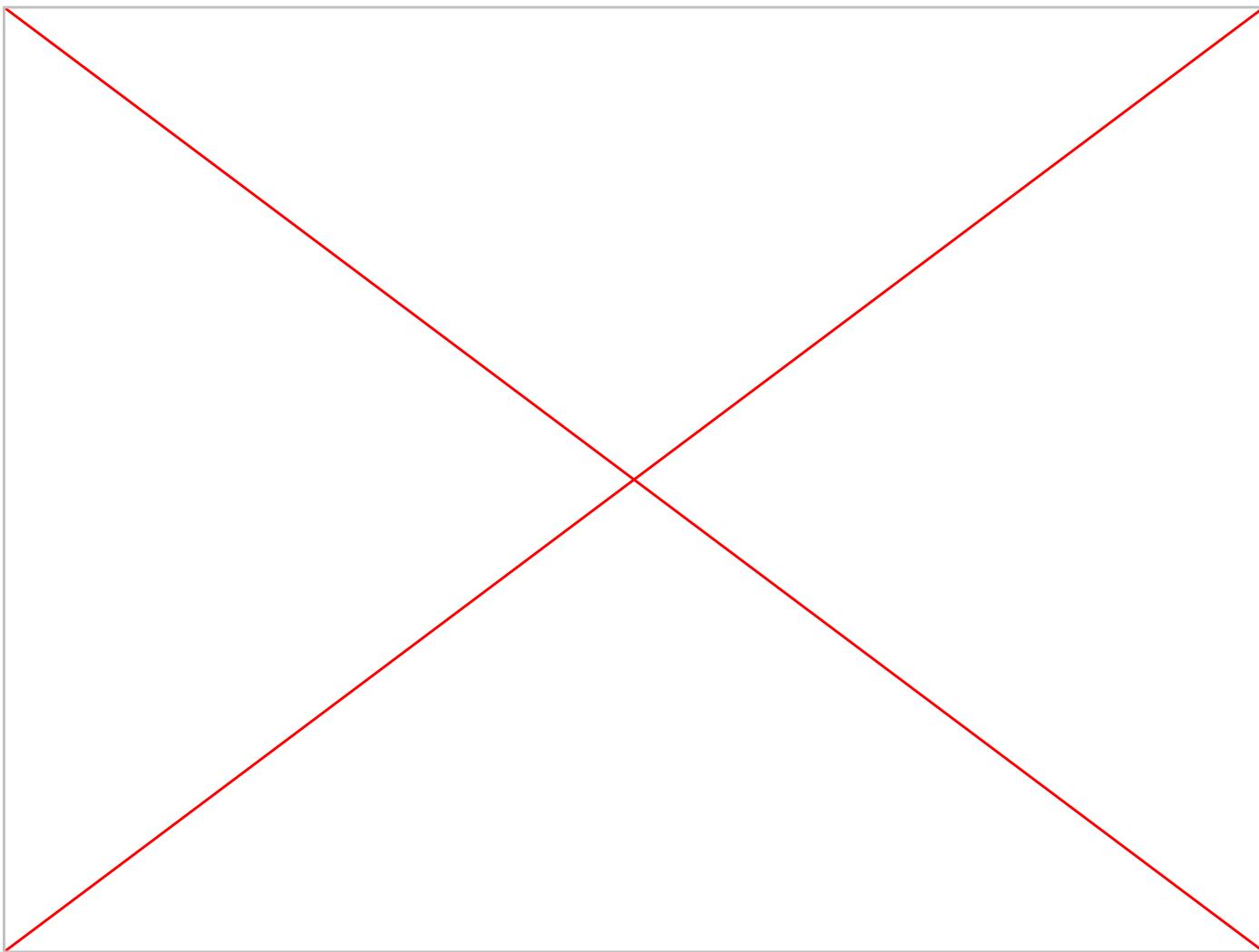


Support Vector Machine

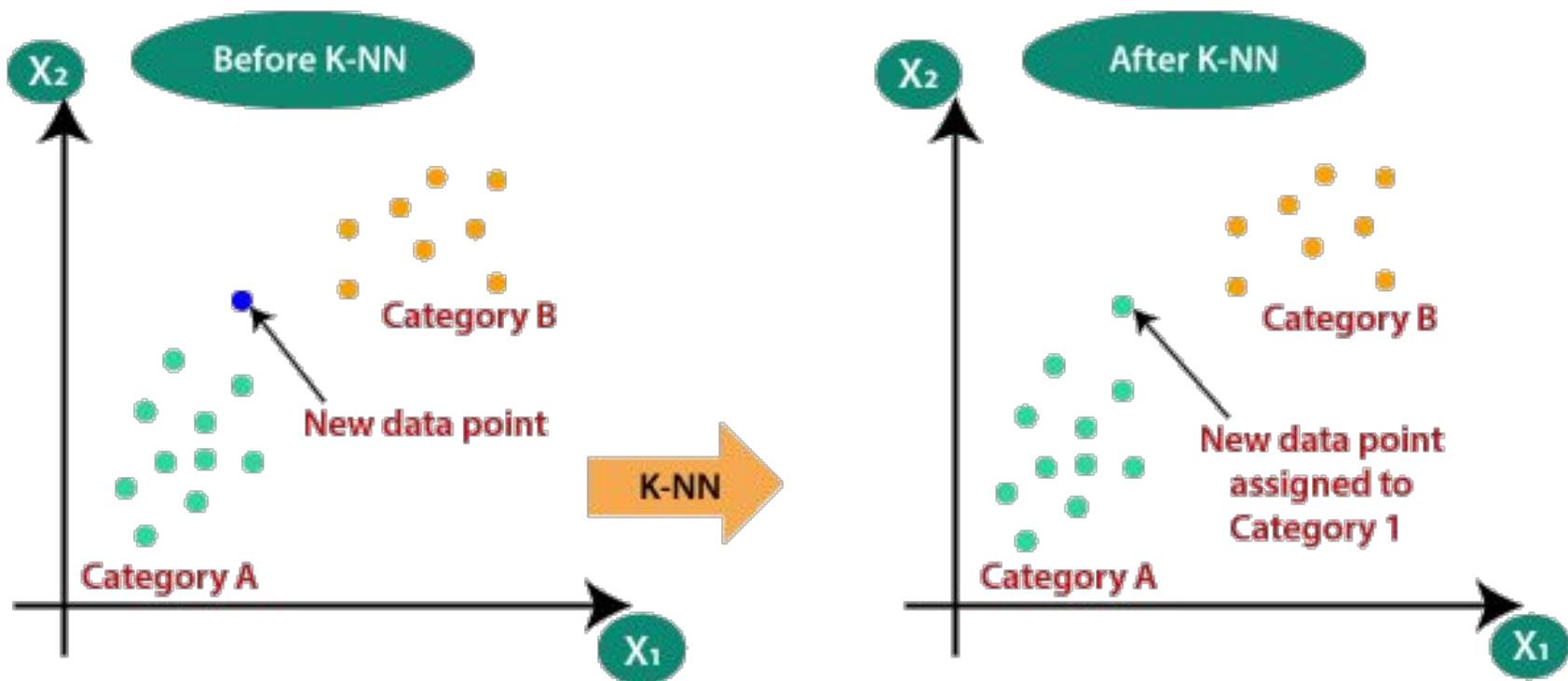


Support Vector Machine

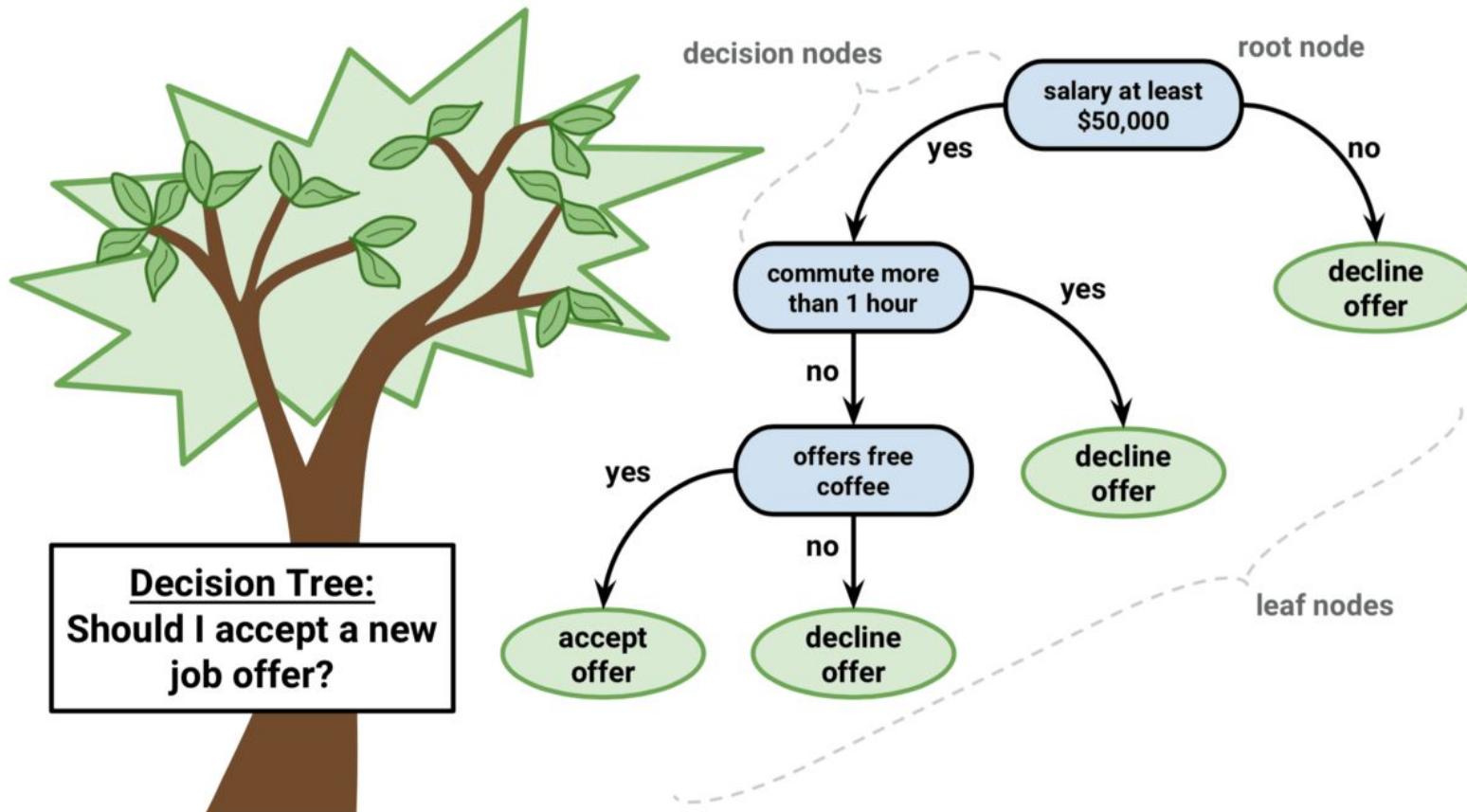




K-Nearest-Neighbour

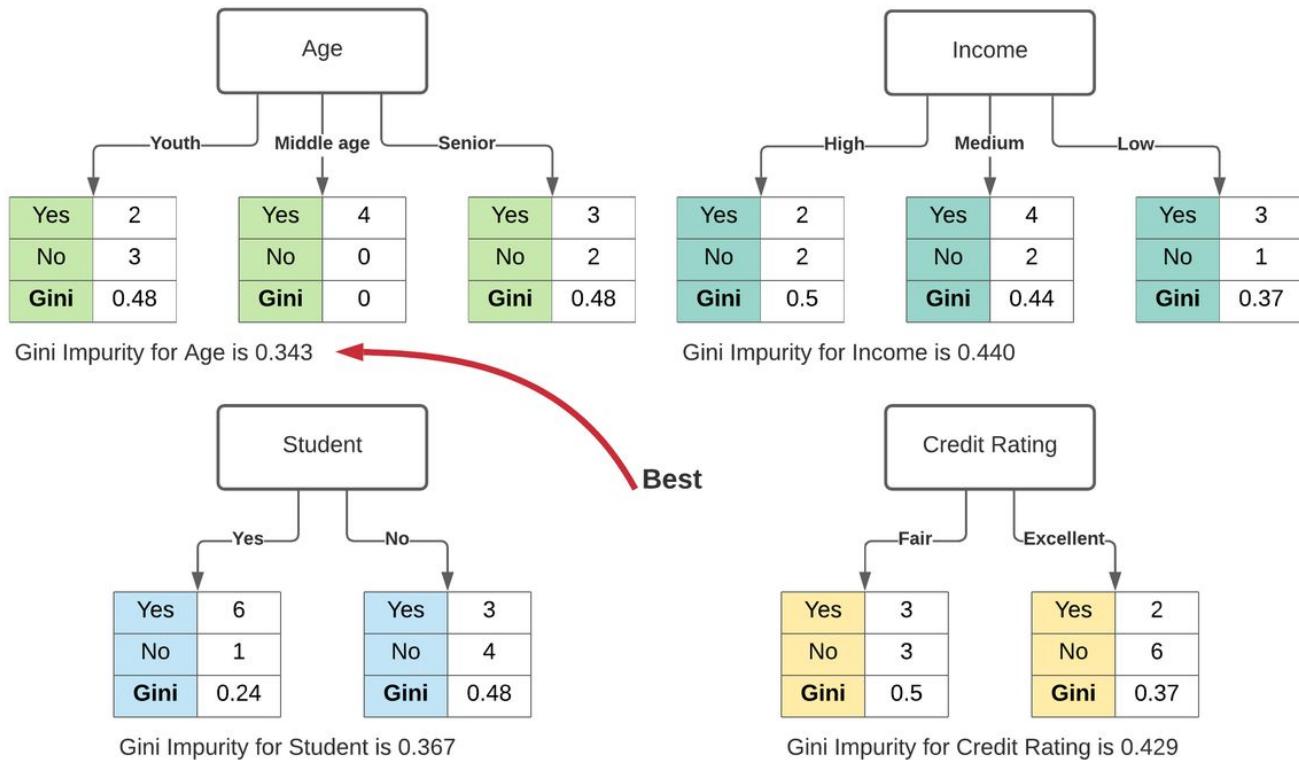


Decision Tree



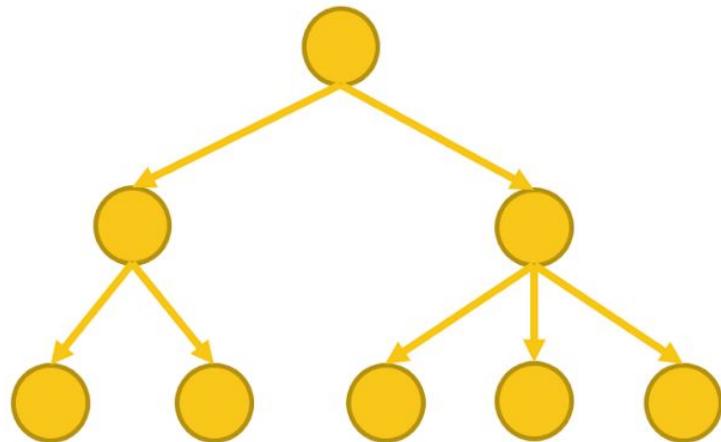
Decision Tree

$$Gini = 1 - \sum_{i=1}^C (p_i)^2$$

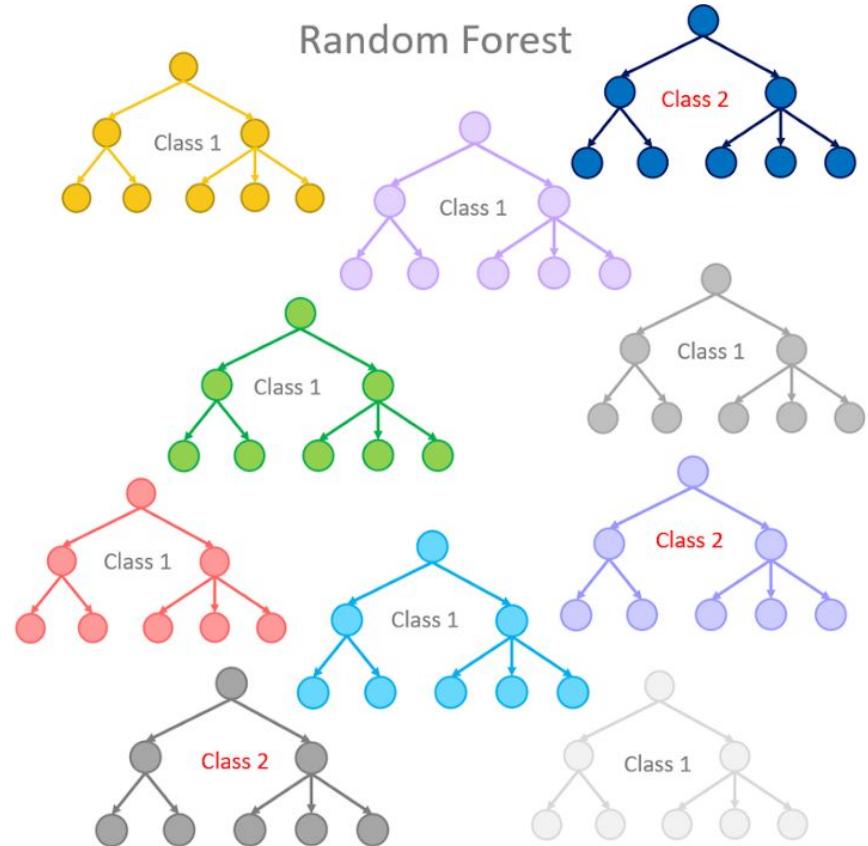


Random Forest

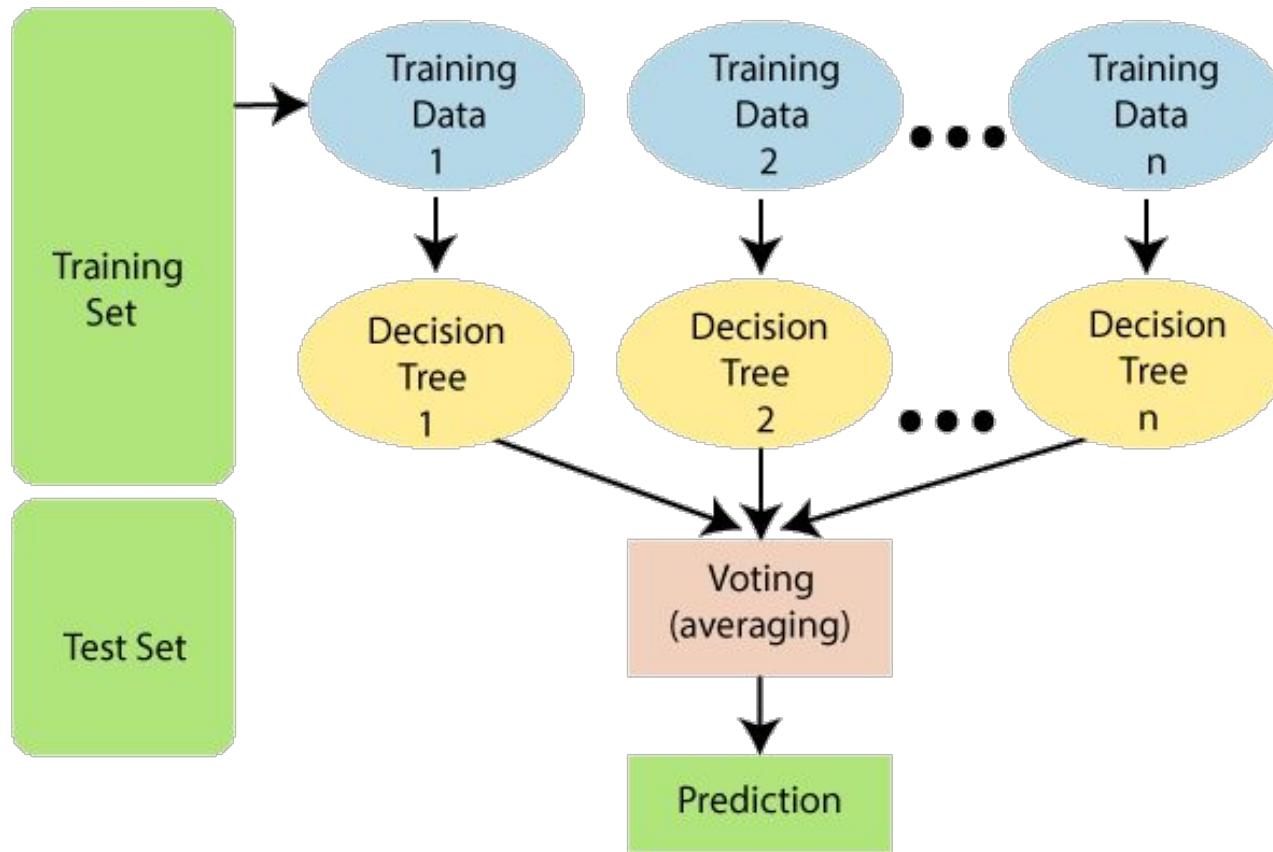
Single Decision Tree



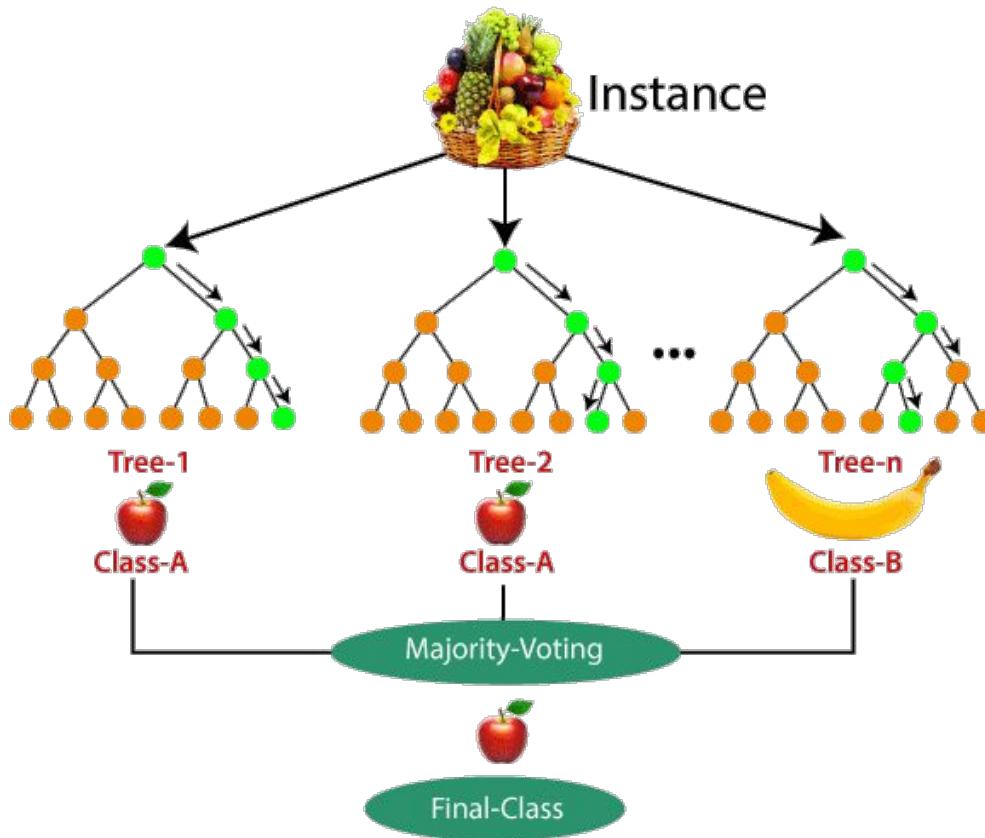
Random Forest



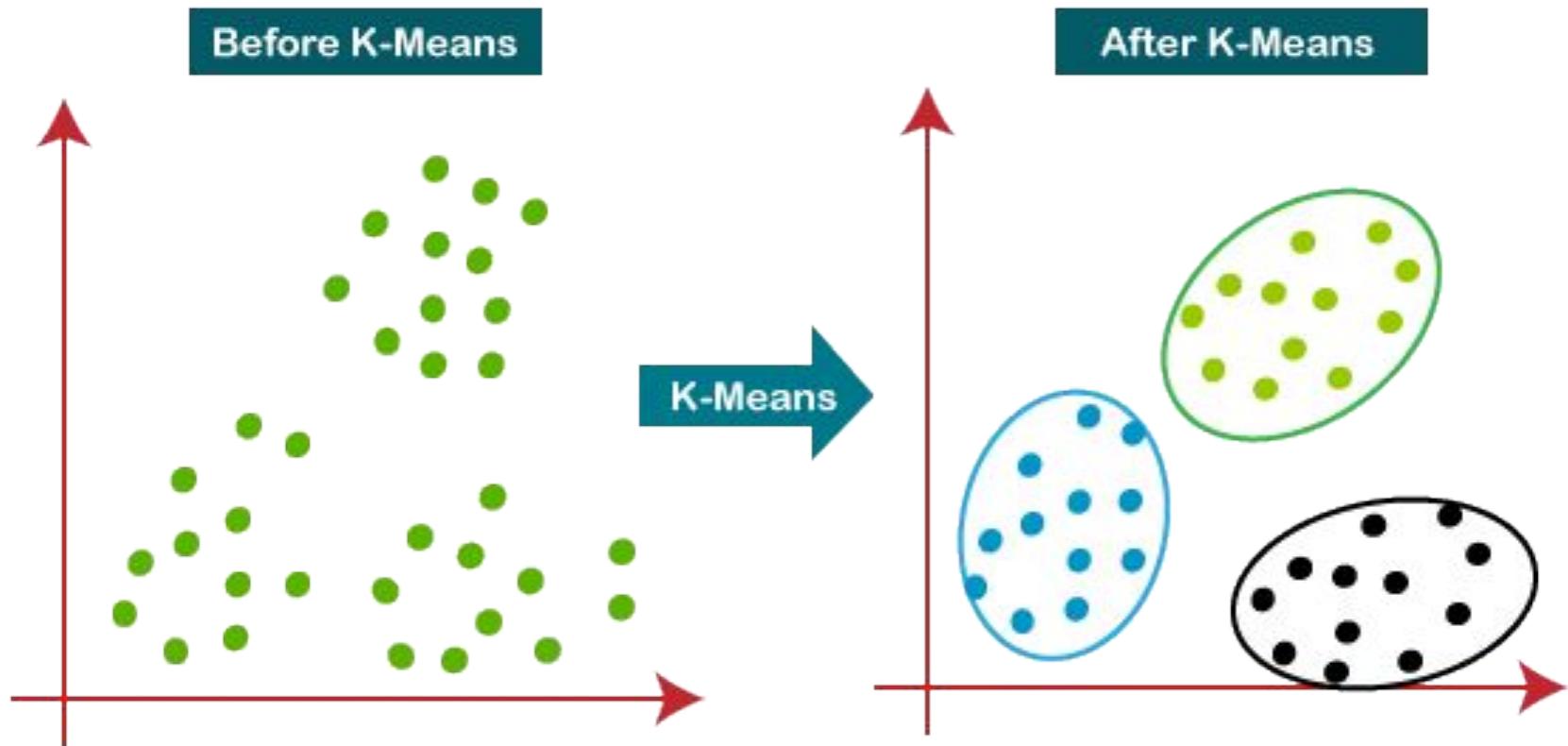
Random Forest

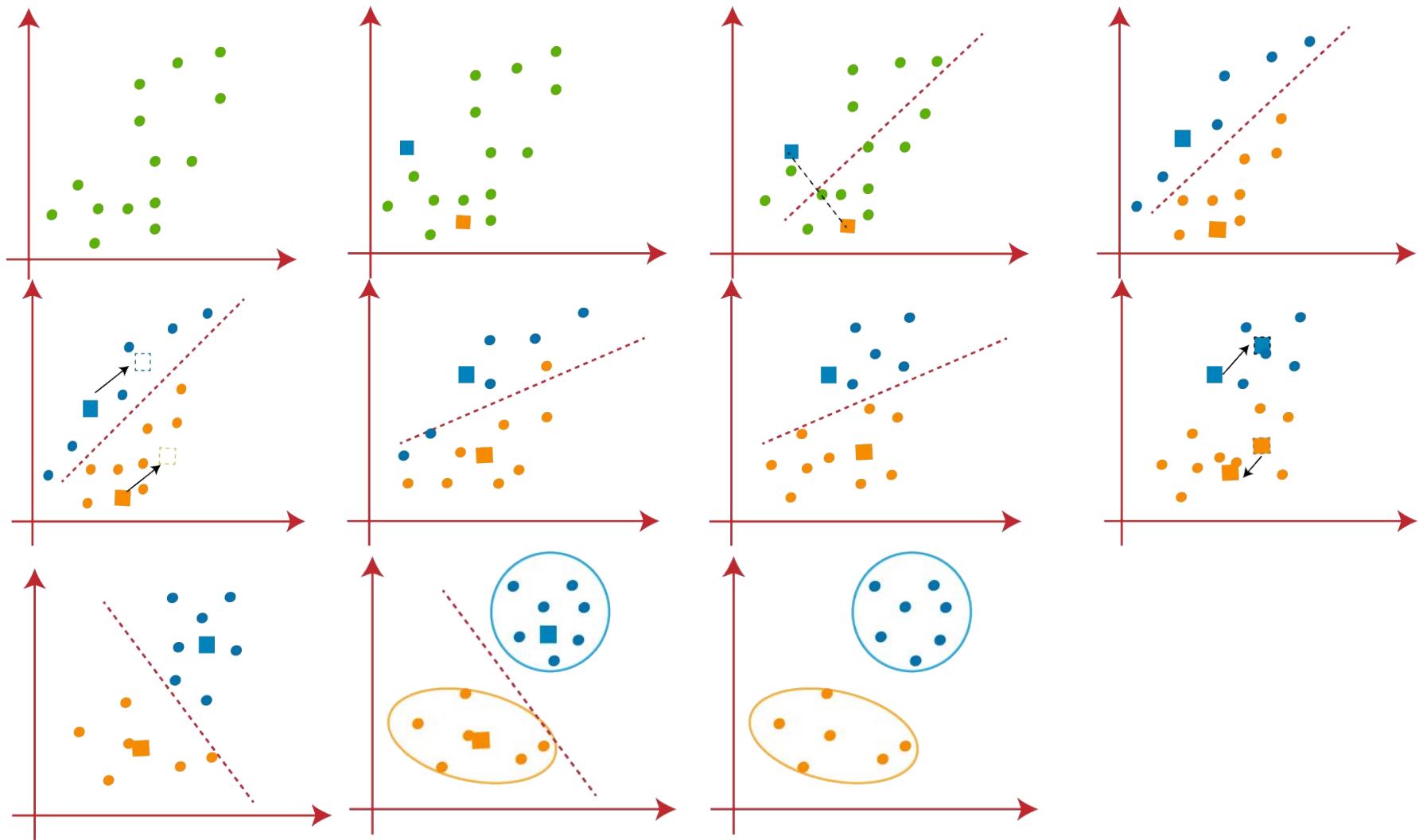


Random Forest

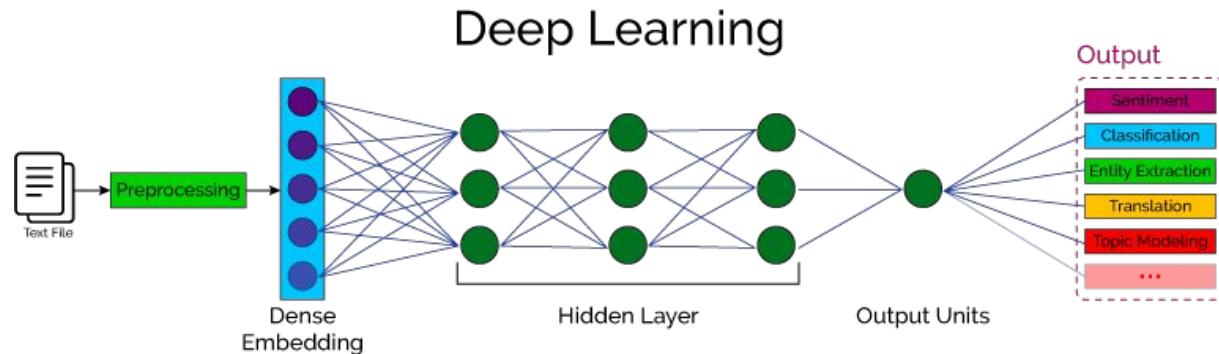
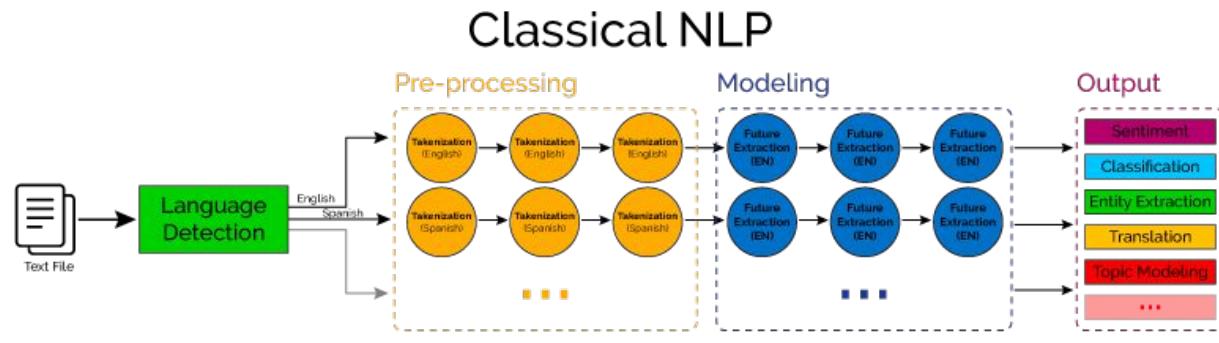
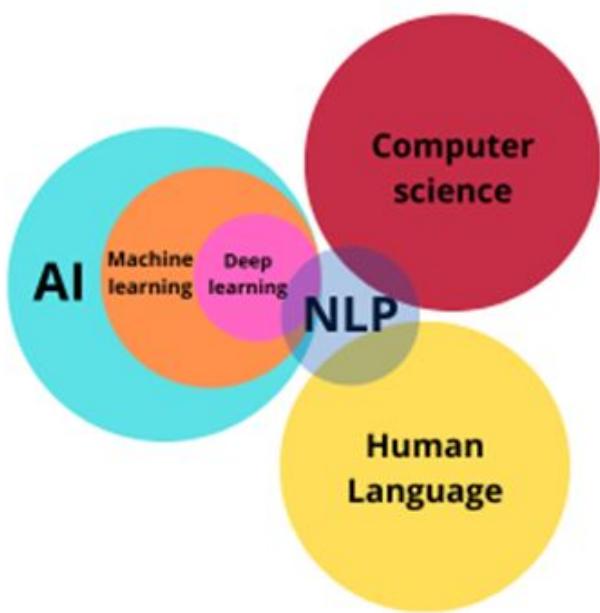


Unsupervised Learning algorithms





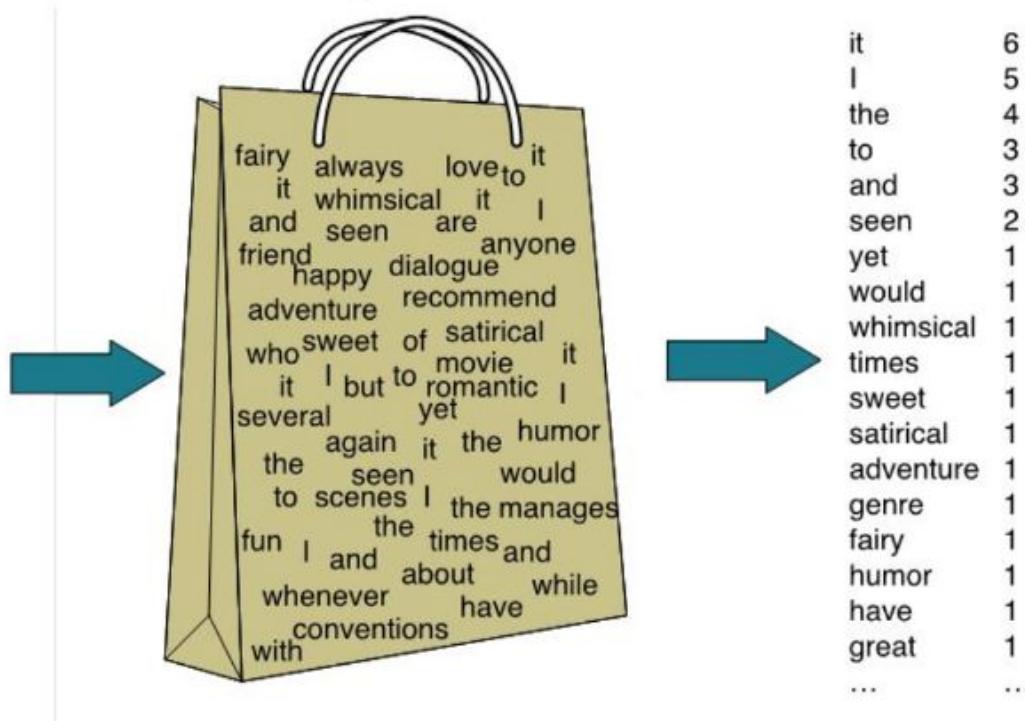
Natural Language Processing



Natural Language Processing

The Bag of Words Representation

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!



Bag of words

```
corpus = [  
    "This is the best book I have ever read",  
    "Reading books is the best way to gain knowledge",  
    "Which book you prefer. This one or the another one?",  
    "Is this the book you love most?"  
]
```

Make vocabulary

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
another	best	book	books	ever	gain	have	is	knowledge	love	most	one	or	prefer	read	reading	the	this	to	way	which	you

Score assignment

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
0	1	1	0	1	0	1	1	0	0	0	0	0	0	1	0	1	1	0	0	0	0
0	1	0	1	0	1	0	1	0	0	0	0	0	0	0	1	1	0	1	1	0	0
1	0	1	0	0	0	0	0	0	0	2	1	1	0	0	1	1	0	0	1	1	0
0	0	1	0	0	0	0	1	0	1	1	0	0	0	0	0	1	1	0	0	0	1

CountVectorizer

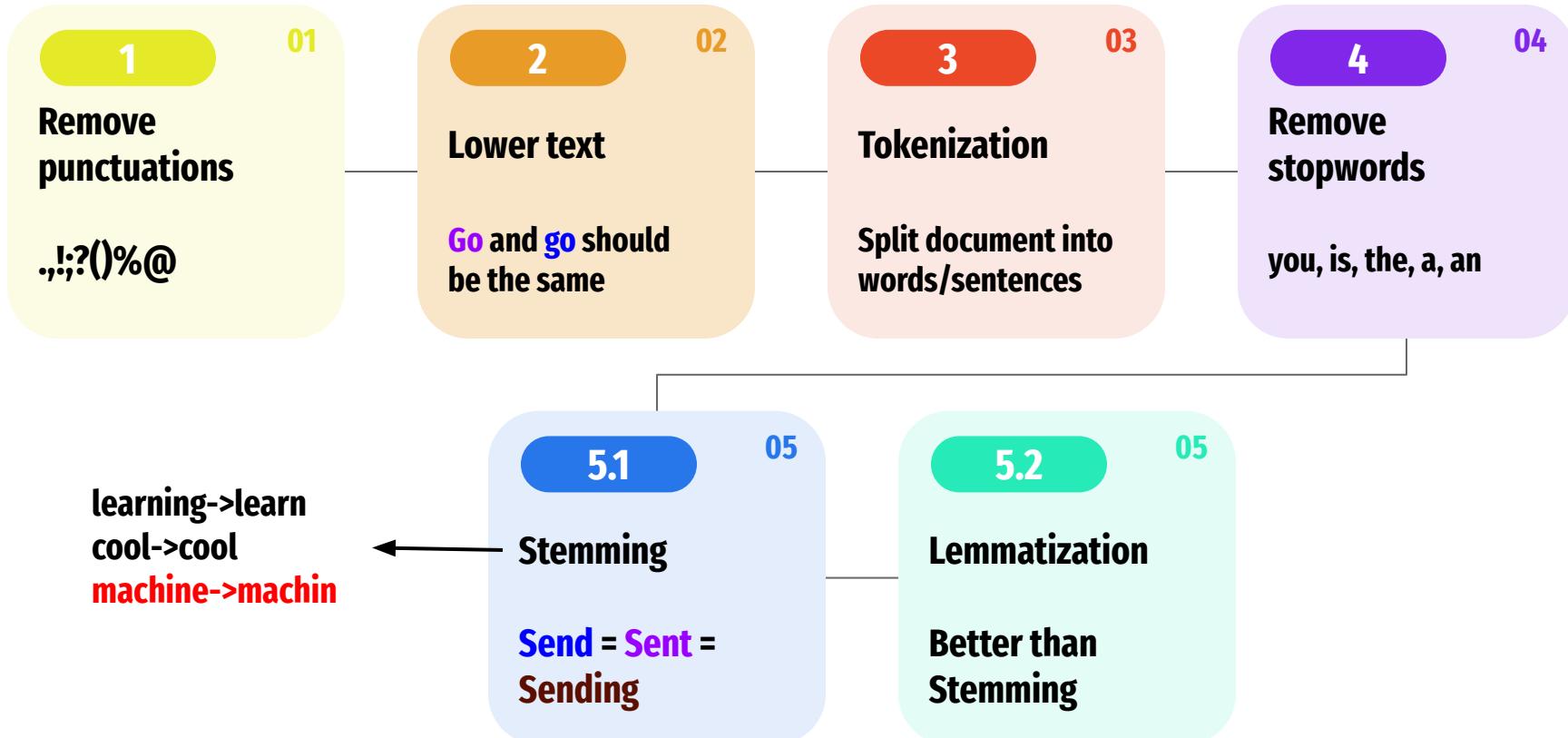
```
from sklearn.feature_extraction.text import CountVectorizer

corpus = [
    "This is the best book I have ever read",
    "Reading books is the best way to gain knowledge",
    "Which book you prefer. This one or the another one?",
    "Is this the book you love most?"
]

vectorizer = CountVectorizer()
data = vectorizer.fit_transform(corpus)
vocabulary = vectorizer.get_feature_names_out()
print("Vocabulary:")
print(vocabulary)
result = data.toarray()
print("Bag of words:")
print(result)
```

```
example ×
C:\Users\Viet\anaconda3\envs\basic\python.exe C:\Users\Viet\PycharmProjects\basic\example.py
Vocabulary:
['another' 'best' 'book' 'books' 'ever' 'gain' 'have' 'is' 'knowledge'
 'love' 'most' 'one' 'or' 'prefer' 'read' 'reading' 'the' 'this' 'to'
 'way' 'which' 'you']
Bag of words:
[[0 1 1 0 1 0 1 1 0 0 0 0 0 0 1 0 1 1 0 0 0 0]
 [0 1 0 1 0 1 0 1 1 0 0 0 0 0 0 1 1 0 1 1 0 0]
 [1 0 1 0 0 0 0 0 0 0 2 1 1 0 0 1 1 0 0 1 1]
 [0 0 1 0 0 0 0 1 0 1 1 0 0 0 0 0 1 1 0 0 0 1]]
```

Text preprocessing



Bag of words

Document	Vocabulary (no stopwords)	Bag of words' vector
This is a good job. I will not ignore it		1,1,1,1,0
This is not a good job. I will ignore it	Good, job, not, ignore, take	1,1,1,1,0
This is a good job. I will take it		1,1,0,0,1

N-grams

N-grams

This is Big Data AI Book

Uni-Gram

This	Is	Big	Data	AI	Book
------	----	-----	------	----	------

Bi-Gram

This is	Is Big	Big Data	Data AI	AI Book
---------	--------	----------	---------	---------

Tri-Gram

This is Big	Is Big Data	Big Data AI	Data AI Book
-------------	-------------	-------------	--------------

CountVectorizer

Term Frequency-Inverse Document Frequency

$$TF = \frac{\text{Number of times a word "X" appears in a Document}}{\text{Number of words present in a Document}}$$

$$IDF = \log \left(\frac{\text{Number of Documents present in a Corpus}}{\text{Number of Documents where word "X" has appeared}} \right)$$

$$TF \cdot IDF = TF * IDF$$

Term Frequency (TF)

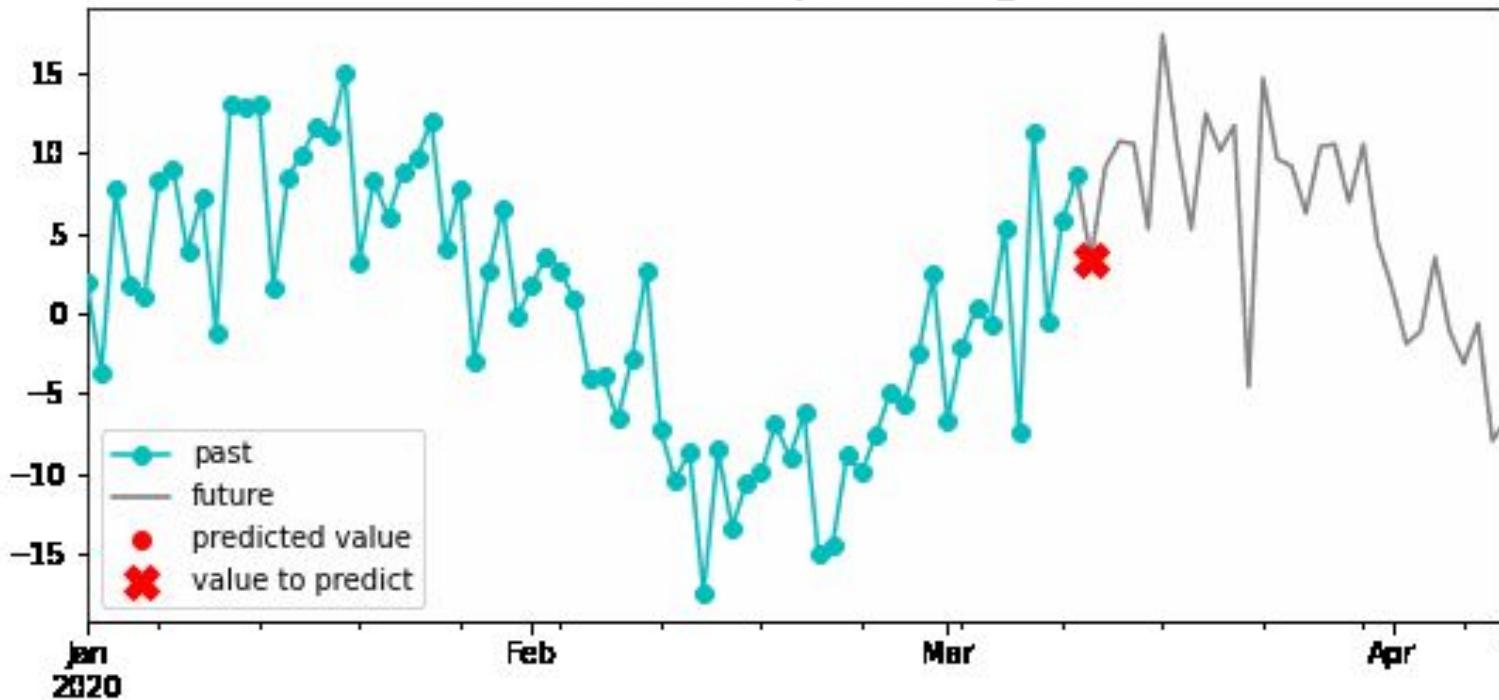
Tần số xuất hiện của **1 word trong 1 document**

Inverse Document Frequency (IDF)

Mức độ phổ biến/hiếm của **1 word trong toàn bộ các documents**

Time series forecasting

Recursive multi-step forecasting: (t+1)



Time series data

date	charter_foreign	total_foreign	charter Domestic	total Domestic
date	bigint	bigint	bigint	bigint
Date	Integer	Integer	Integer	Integer
2008-01-01T00:00:00.000Z	57463	5642057	174692	7394181
2008-02-01T00:00:00.000Z	56651	5053435	142931	6837201
2008-03-01T00:00:00.000Z	95878	6071551	176906	8439191
2008-04-01T00:00:00.000Z	100198	5731709	103054	7467710
2008-05-01T00:00:00.000Z	130720	6003335	88631	7840260
2008-06-01T00:00:00.000Z	196656	6467543	99579	8274573
2008-07-01T00:00:00.000Z	195873	6861571	130766	8971734
2008-08-01T00:00:00.000Z	152848	6895232	77253	8717085
2008-09-01T00:00:00.000Z	88267	5539617	40070	6371669
2008-10-01T00:00:00.000Z	76723	5564387	43003	6520788
2008-11-01T00:00:00.000Z	26671	5128350	50322	6223857
2008-12-01T00:00:00.000Z	30963	5624131	54894	6959043
2009-01-01T00:00:00.000Z	35174	5399479	60652	6805171

A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	Jan-17	Feb-17	Mar-17	Apr-17	May-17	Jun-17	Jul-17	Aug-17	Sep-17	Oct-17	Nov-17	Dec-17	
2 Store 1	20 809	23 832	22 542	18 736	15 347	12 190	11 049	10 913	9 019	9 803	3 172	7 768	
3 Store 2	14 834	14 170	13 109	7 560	6 388	8 124	6 719	3 847	3 187	8 145	10 007	10 036	
4 Store 3	14 453	12 816	14 260	12 954	12 596	11 200	8 374	9 725	10 853	11 609	13 897	19 628	
5 Store 4	8 590	8 822	13 497	13 073	13 278	14 705	13 550	13 962	14 783	11 712	13 234	17 790	
6 Store 5	16 647	12 851	13 433	13 486	13 767	10 482	11 476	13 024	14 459	14 672	12 885	16 681	
7 Store 6	7 323	5 043	4 857	2 584	6 990	4 130	45 110	4 895	5 708	5 106	4 334	10 741	
8 Store 7	8 735	4 645	4 905	4 536	5 385	5 040	6 341	9 403	9 904	9 521	13 473	7 077	
9 Store 8	965	1 254	1 937	1 940	1 756	1 690	1 536	2 502	2 806	4 087	3 704	5 478	
10 Store 9	188	4 375	5 109	5 255	5 667	6 331	9 182	9 867	8 948	10 096	10 860	10 951	
11 Store 10	14 033	30 900	18 864	12 828	10 269	8 464	7 674	4 160	5 497	9 608	16 177	15 798	
12 Store 11	8 185	9 090	12 560	13 682	10 061	8 172	8 045	6 622	7 240	9 923	9 403	10 768	
13 Store 12	911	384	658	949	1 236	1 405	1 366	1 433	1 282	1 515	1 457	2 933	
14 Store 13	2 361	2 830	2 892	2 953	2 929	2 982	6 361	25 969	22 952	27 171	27 199	35 298	
15 Store 14	9 805	8 436	12 878	9 580	6 665	5 069	6 003	6 139	10 987	11 039	11 252	13 844	
16													

A	B	C	D	E	F	G	
1	Code	Name	Population 1990	Population 2000	Population 2010	Population Change 1980-1990	Population Change 1990-2000
2 AK	Alaska		550043	626932	710231	36.9	14
3 AL	Alabama		4040587	4447100	4779736	3.8	10.1
4 AR	Arkansas		2350725	2673400	2915918	2.8	13.7
5 AZ	Arizona		3665228	5130632	6392017	34.8	40
6 CA	California		29760021	33871648	37253956	25.7	13.8
7 CO	Colorado		3294394	4301261	5029196	14	30.6
8 CT	Connecticut		3287116	3405565	3574097	5.8	3.6
9 DC	District of Columbia		606900	572059	601723	-4.9	-5.7
10 DE	Delaware		666168	783600	897934	12.1	17.6
11 FL	Florida		12937926	15982378	18801310	32.7	23.5
12 GA	Georgia		6478216	8186453	9687653	18.6	26.4
13 HI	Hawaii		1108229	1211537	1360301	14.9	9.3
14 IA	Iowa		2776755	2926324	3046355	-4.7	5.4
15 ID	Idaho		1006749	1293953	1567582	6.7	28.5
16 IL	Illinois		11430602	12419293	12830632	0	8.6
17 IN	Indiana		5544159	6080485	6483802	1	9.7
18 KS	Kansas		2477574	2688418	2853118	4.8	8.5
19 KY	Kentucky		3685296	4041769	4339367	0.7	9.7
20 LA	Louisiana		4719973	4468976	4533377	0.3	5.9

Time series data

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

Time series

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

Exogenous variable

a	b	c	d	e	f	g	h	i	j
---	---	---	---	---	---	---	---	---	---

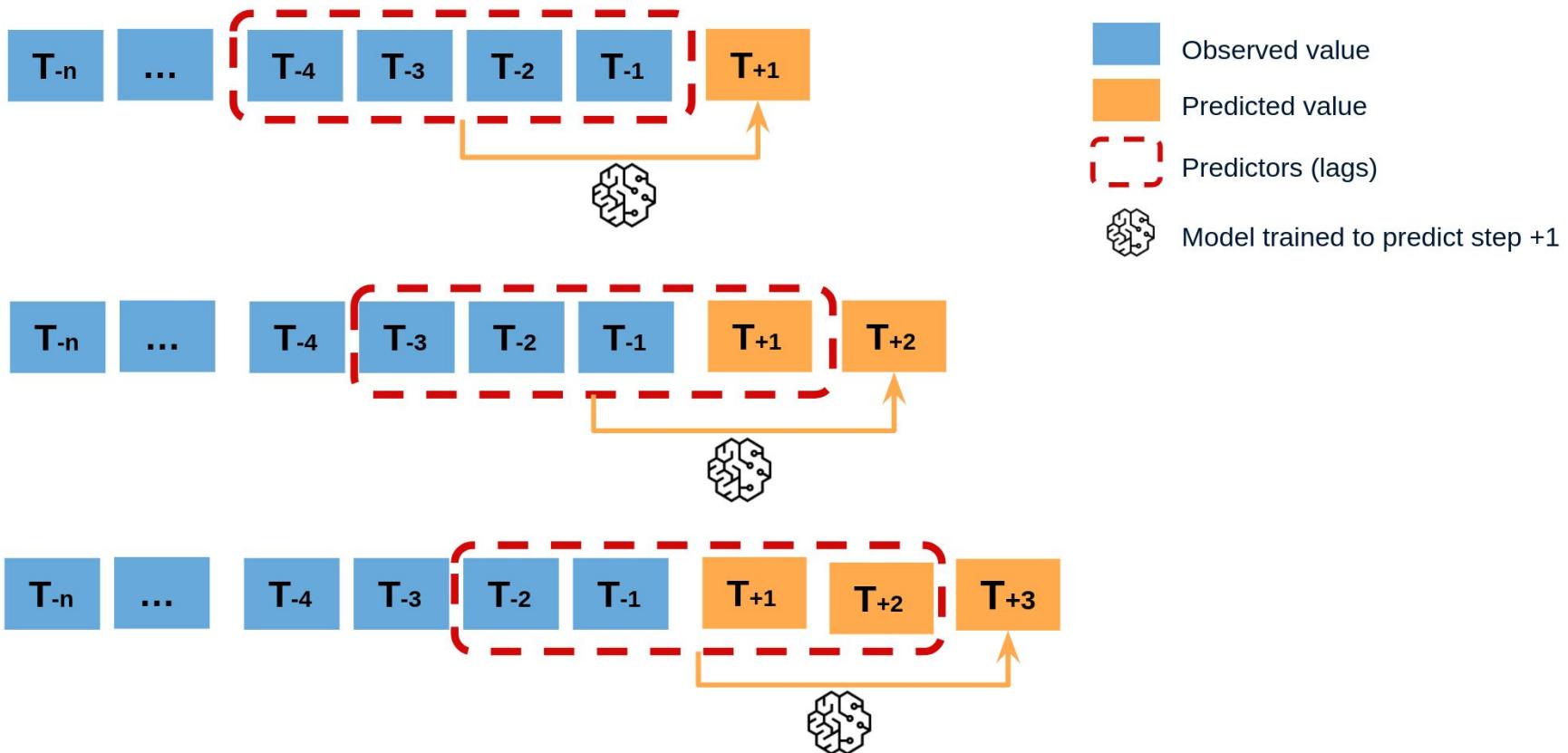
X

1	2	3	4	5	f
2	3	4	5	6	g
3	4	5	6	7	h
4	5	6	7	8	i
5	6	7	8	9	j

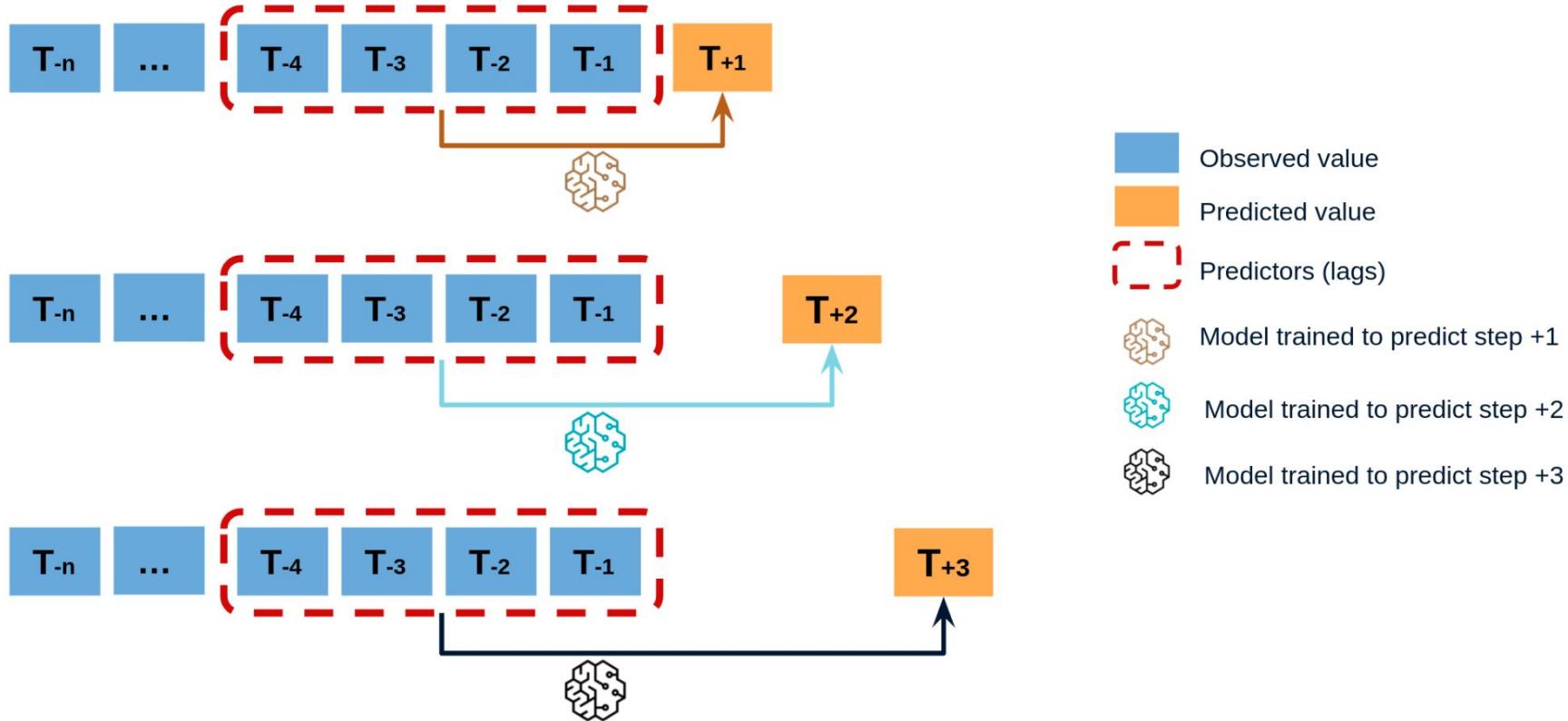
y

6
7
8
9
10

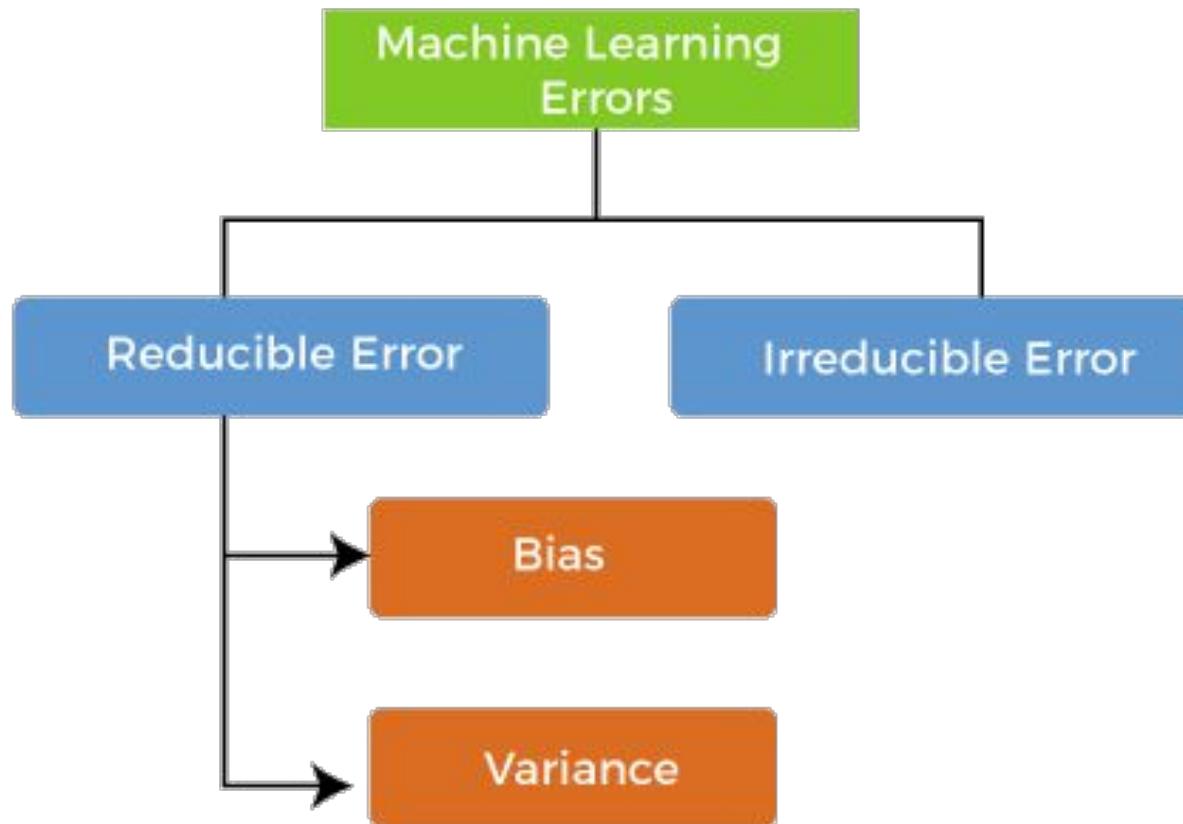
Recursive multi-step Time series Forecasting



Direct multi-step Time series Forecasting



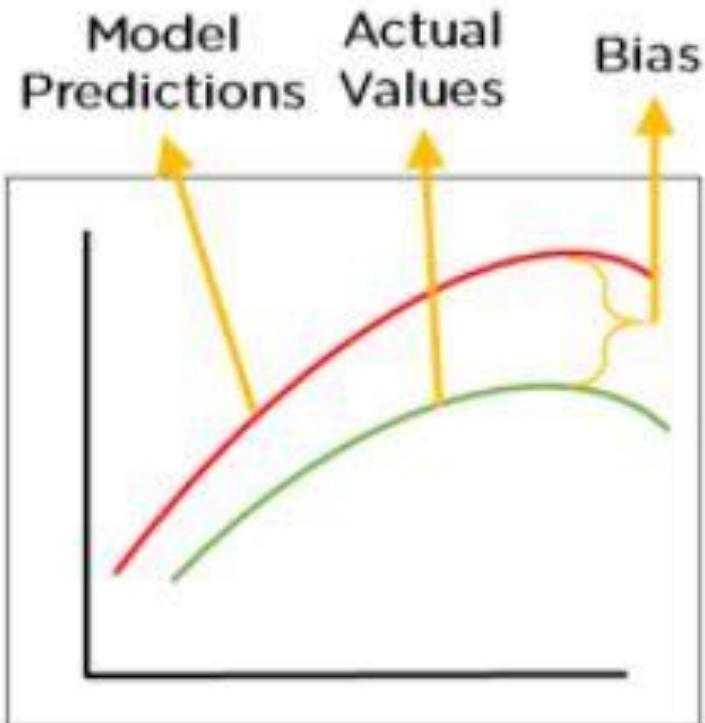
Error in Machine Learning



Error in Machine Learning

Bias

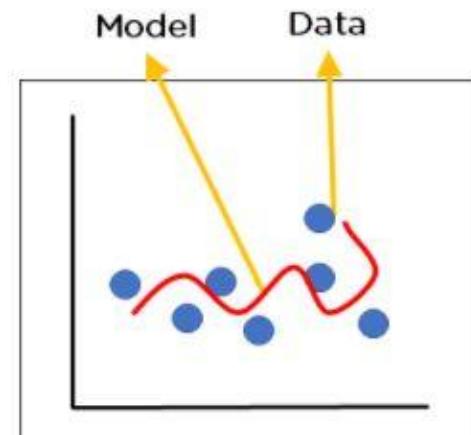
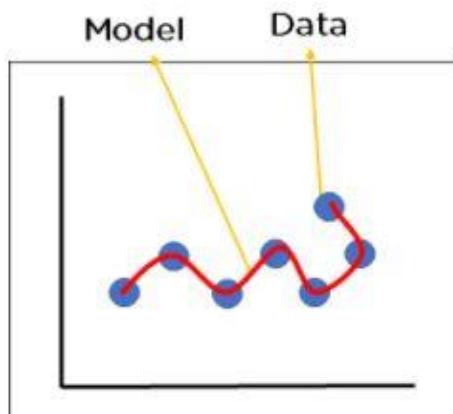
- Sai lệch giữa prediction và actual label
- Cho ta biết khả năng dự đoán chính xác của mô hình
- Bias càng bé càng tốt
- High bias means:
 - Overly-simplified model
 - Under-fitting
 - High error on both training and test sets



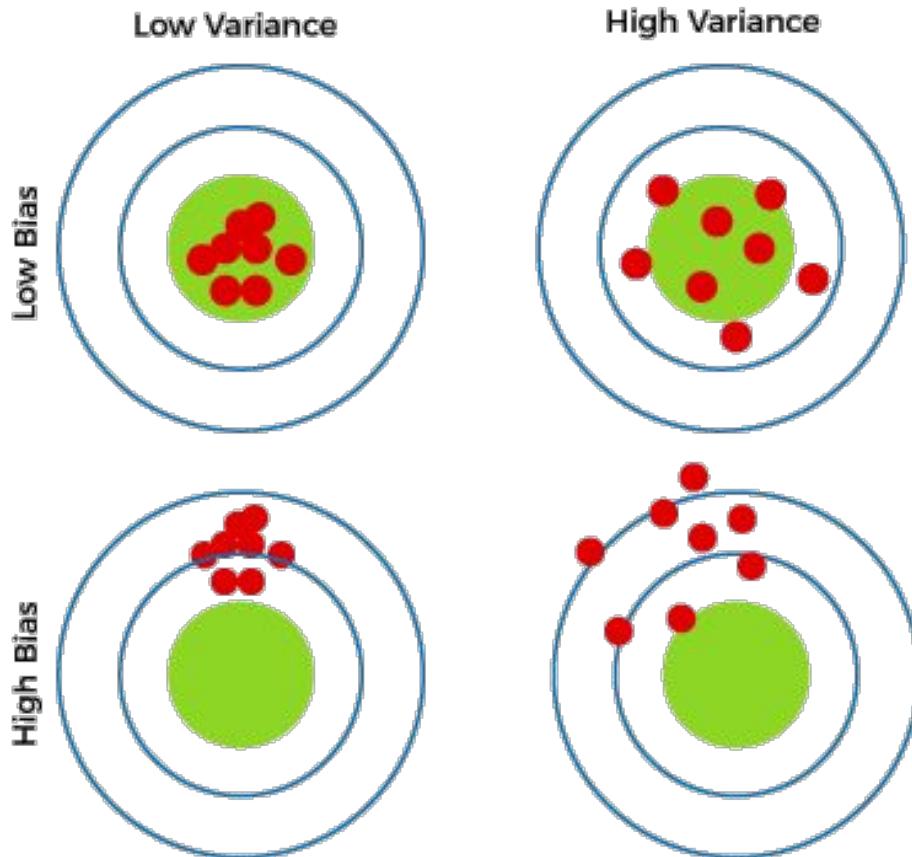
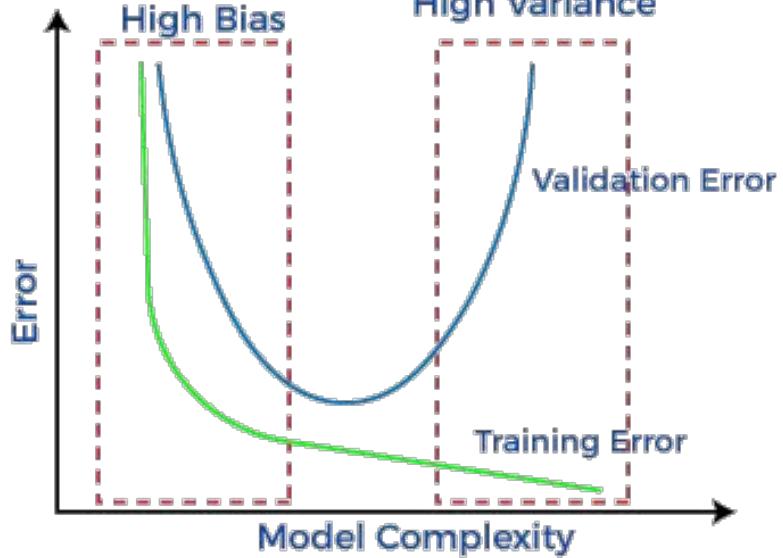
Error in Machine Learning

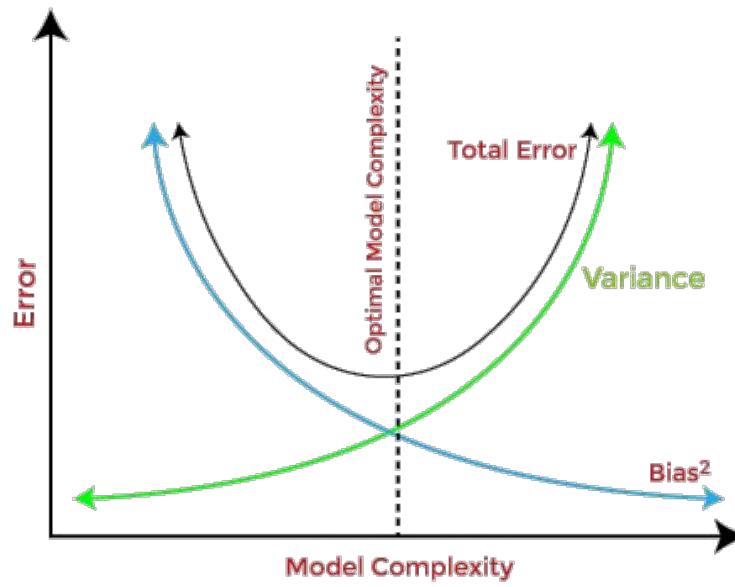
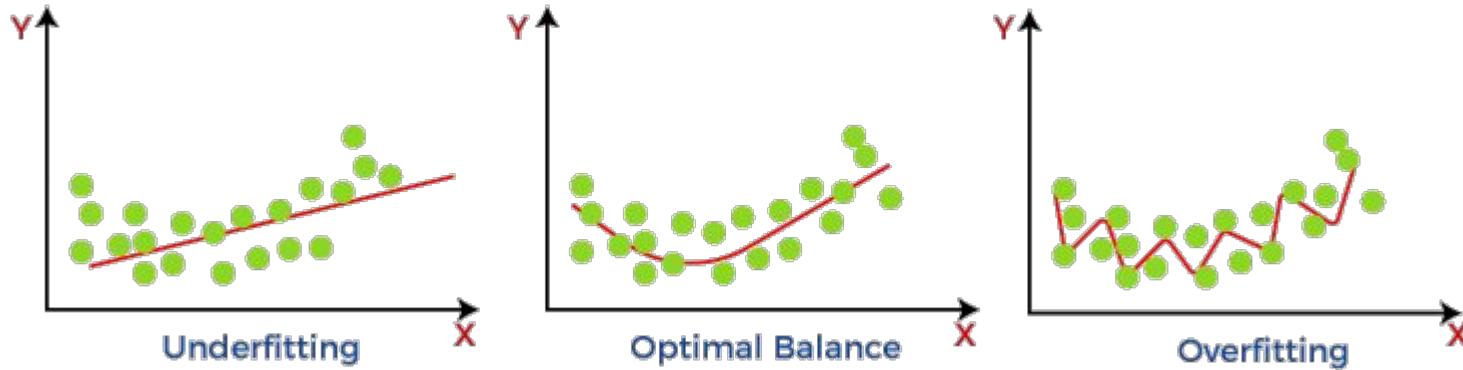
Variance

- Mức độ thay đổi của độ chính xác của prediction nếu input thay đổi
- Cho ta biết khả năng tổng quát hóa của mô hình
- Variance càng bé càng tốt
- High variance means:
 - Overly-complex model
 - Over-fitting
 - Low error on training set but high error on test set



Bias-Variance tradeoff





Choose Machine Learning model based on data

Low bias + high variance

- Decision tree
- Random Forest
- K-nearest neighbours
- Kernel SVM

-> Phù hợp khi có **nhiều data** và **ít features**

High bias + low variance

- Linear regression
- Logistic regression
- Linear SVM

-> Phù hợp khi có **ít data** và **nhiều features**

Extra

Nếu có **nhiều data** và **nhiều features**

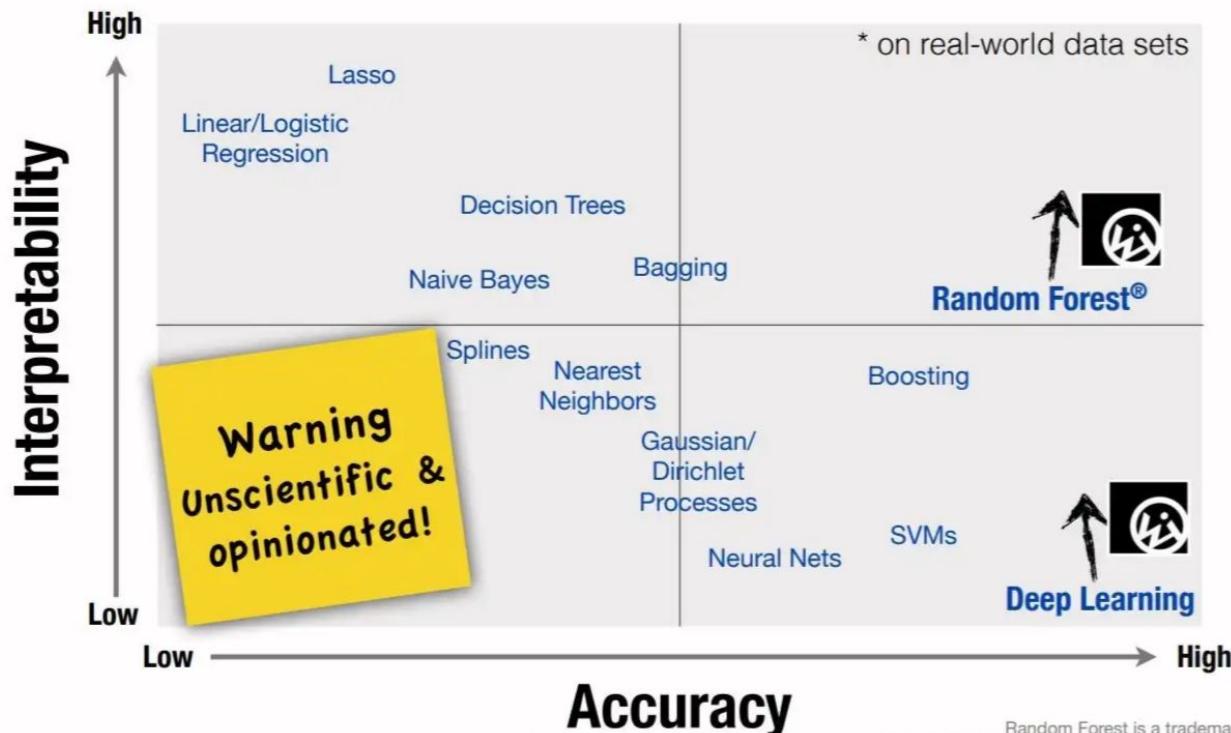
- Giảm chiều dữ liệu (e.g. PCA)
- Sử dụng Neural network

Choose Machine Learning model based on business domain

- **Which metric is important?**
 - Accuracy, precision, recall, F1, ...
- **What is the priority?**
 - Speed: self-driving car app need to be realtime -> fast
 - Memory: Model deployed in embedded device needs to be small
 - Accuracy: In medical field, accuracy is the most important factor
- **Interpretability ?**
 - Do you need to explain result?
 - Do you need to find out which features are important?

Machine Learning algorithm's interpretability

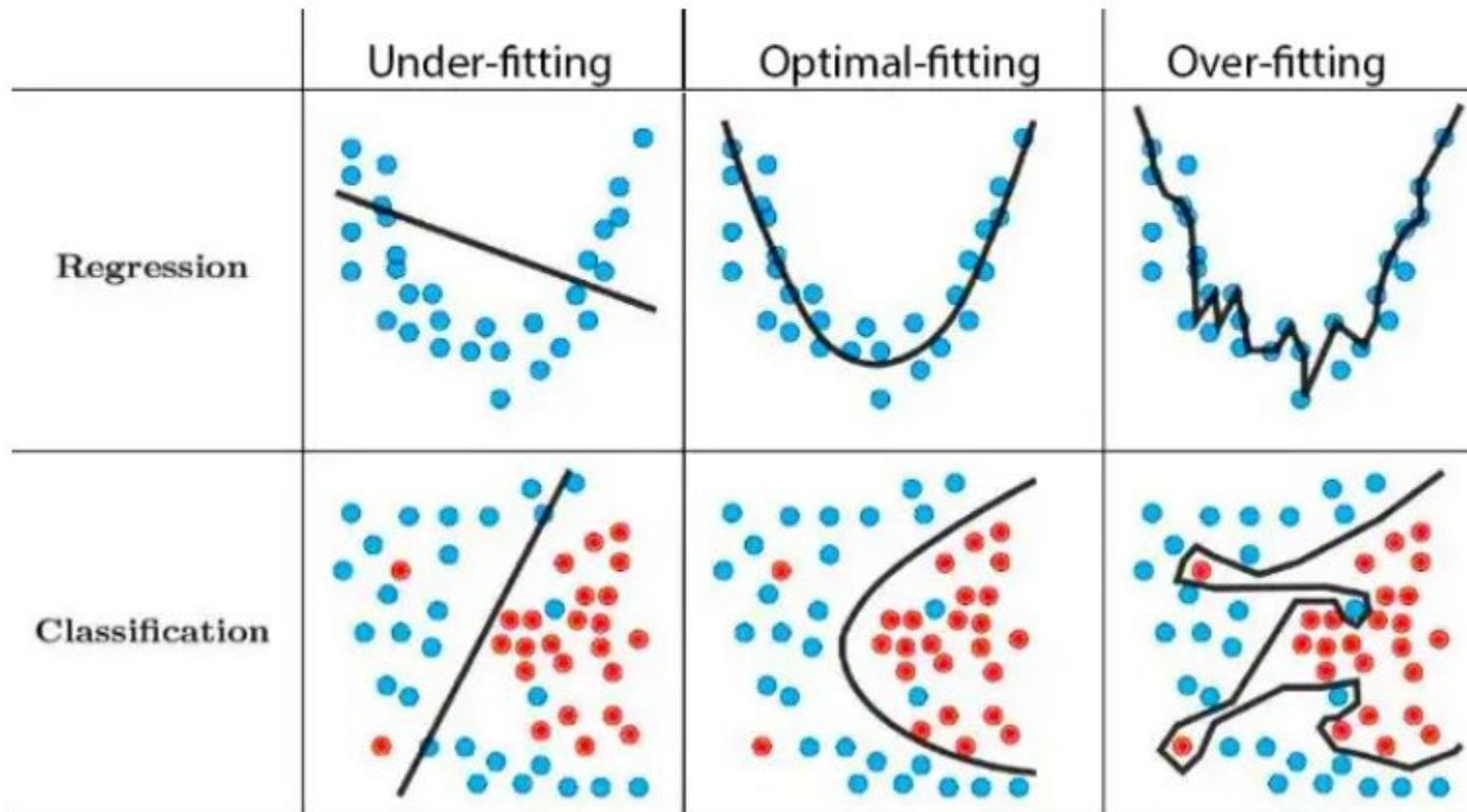
ML Algorithmic Trade-Off



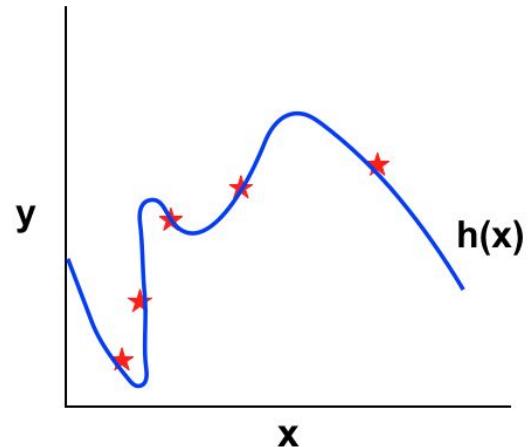
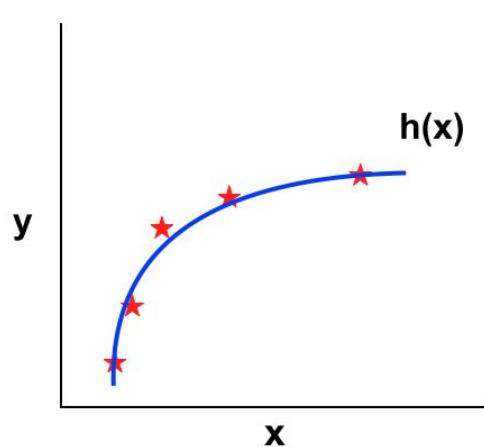
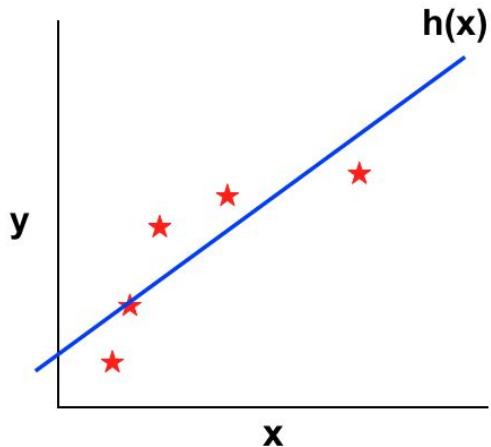
Tips for choosing Machine Learning model

- **Start with a simple model**
 - Choose the simplest model first
 - If it is good enough, you even do not need to try another model
- **Try different models and shortlist the best ones?**
- **Do Hyperparameter Tuning for each models?**
 - GridSearchCV if the number of combinations is small
 - RandomizedSearchCV if the number of combination is large
- **Compare amongst the best models with best hyperparameters to pick up the best one**

Underfitting and Overfitting



Regularization



$$L(x, y) = \sum_{i=1}^n (y_i - f(x_i))^2$$

$$f(x_i) = h_\theta x = \theta_0 + \theta_1 x_1 + \theta_2 x_2^2 + \theta_3 x_3^3 + \theta_4 x_4^4$$

$$f(x_i) = h_\theta x = \theta_0 + \theta_1 x_1 + \theta_2 x_2^2 + \theta_3 \cancel{x_3^3} + \theta_4 \cancel{x_4^4}$$

$$f(x_i) = h_\theta x = \theta_0 + \theta_1 x_1 + \theta_2 x_2^2$$

Regularization

L1 Regularization

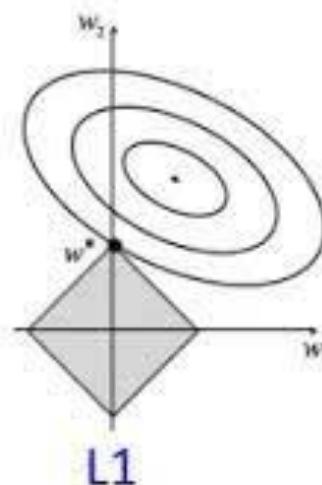
$$\text{Cost} = \sum_{i=0}^N (y_i - \sum_{j=0}^M x_{ij} W_j)^2 + \lambda \sum_{j=0}^M |W_j|$$

L2 Regularization

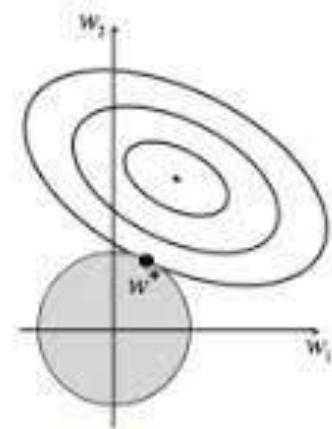
$$\text{Cost} = \sum_{i=0}^N (y_i - \sum_{j=0}^M x_{ij} W_j)^2 + \lambda \sum_{j=0}^M W_j^2$$

Loss function

Regularization
Term

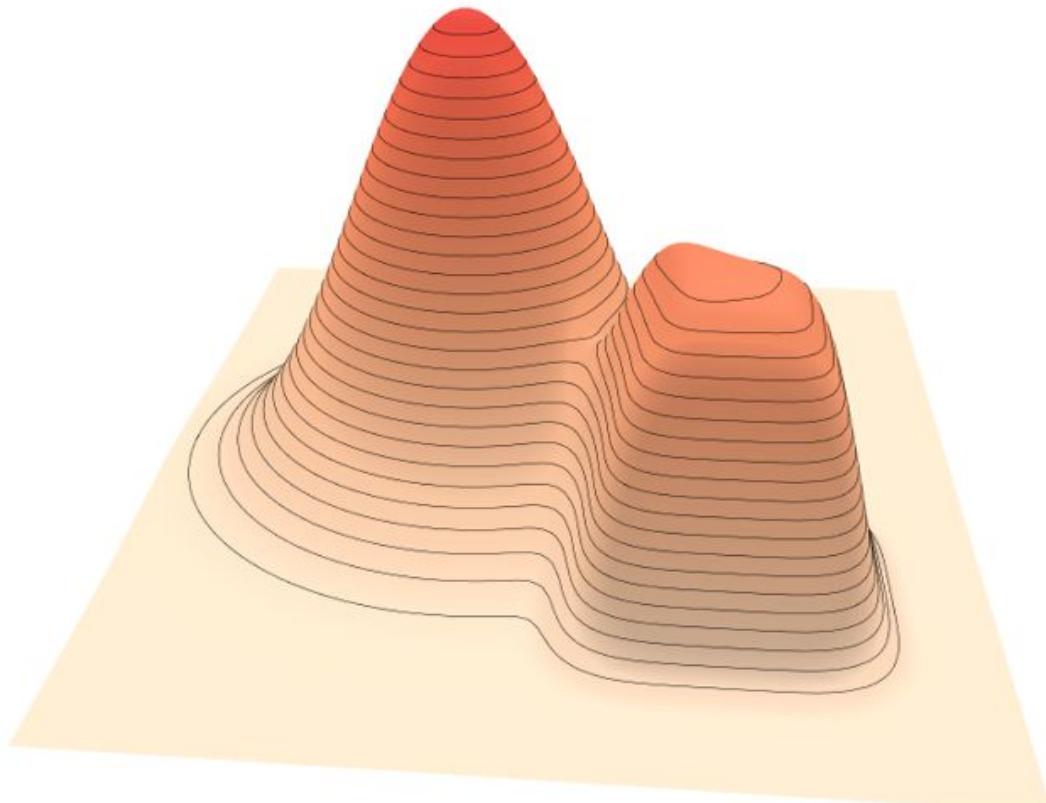


L1

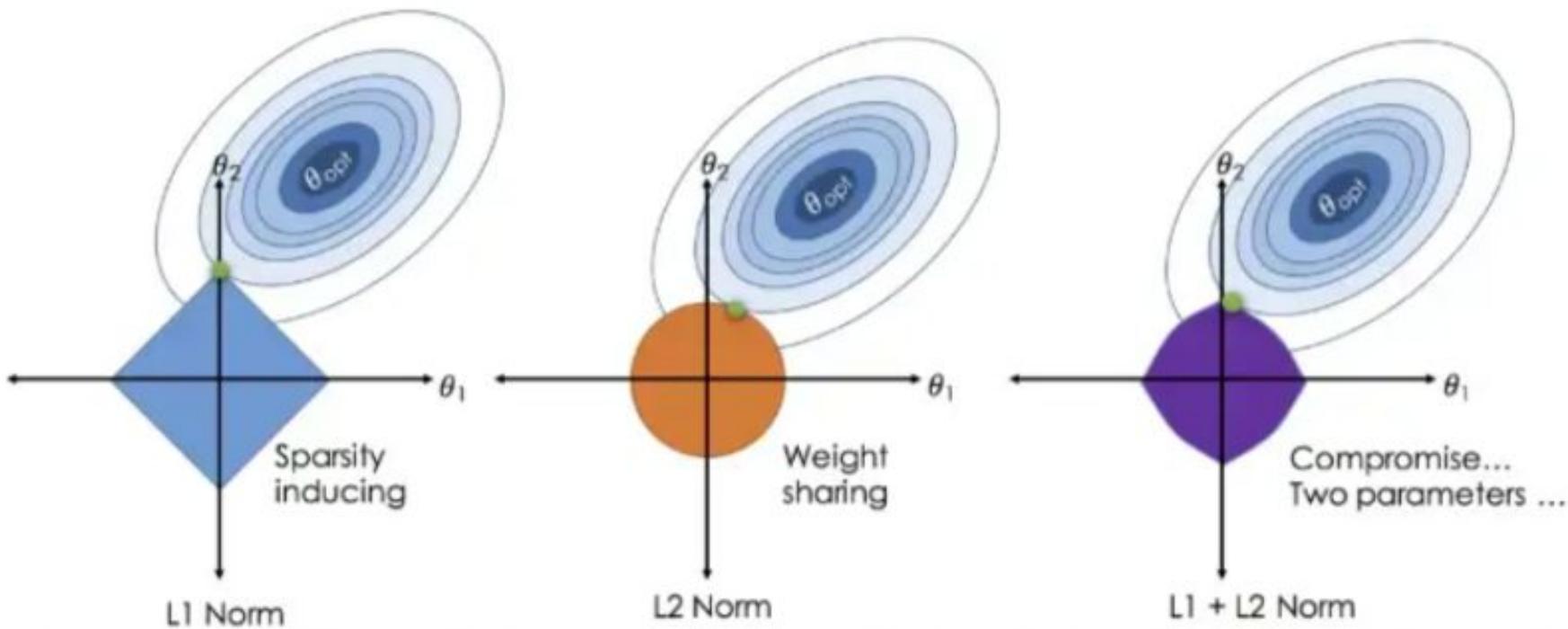


L2

Regularization



Regularization



Regularization Example

```
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline

data = pd.read_csv("diabetes.csv")
x = data.drop("Outcome", axis=1)
y = data["Outcome"]
cls = Pipeline(steps=[
    ('scaler', StandardScaler()),
    ('model', LogisticRegression(penalty="l1", solver="liblinear"))
])
cls.fit(x, y)
for i, j in zip(x.columns, cls["model"].coef_[0]):
    print("{} is with coefficient of {}".format(i, j))
```

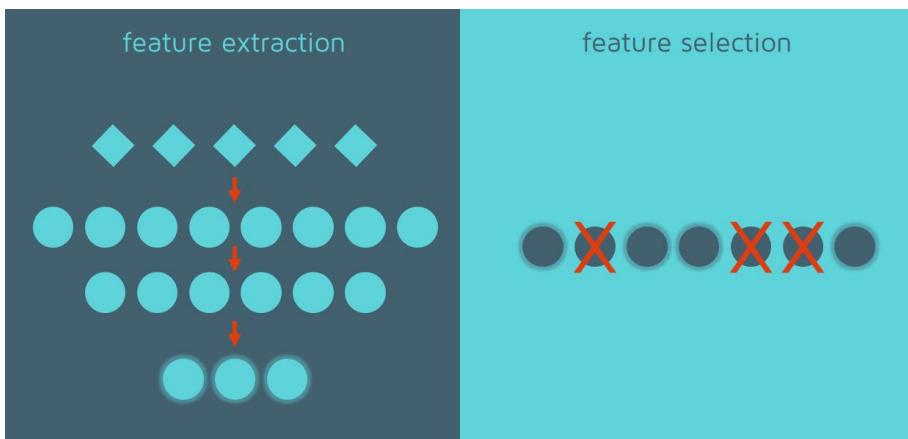
fuel_viz ×
C:\Users\Viet\anaconda3\envs\basic\python.exe C:\Users\Viet\PycharmPr
Pregnancies is with coefficient of 0.4047757715484604
Glucose is with coefficient of 1.1020534150332097
BloodPressure is with coefficient of -0.23905878835657515
SkinThickness is with coefficient of 0.0
Insulin is with coefficient of -0.12014123606887196
BMI is with coefficient of 0.6883338771775394
DiabetesPedigreeFunction is with coefficient of 0.30157785094254597
Age is with coefficient of 0.16766878675897076

```
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline

data = pd.read_csv("diabetes.csv")
x = data.drop("Outcome", axis=1)
y = data["Outcome"]
cls = Pipeline(steps=[
    ('scaler', StandardScaler()),
    ('model', LogisticRegression(penalty="l2", solver="lbfgs"))
])
cls.fit(x, y)
for i, j in zip(x.columns, cls["model"].coef_[0]):
    print("{} is with coefficient of {}".format(i, j))
```

fuel_viz ×
C:\Users\Viet\anaconda3\envs\basic\python.exe C:\Users\Viet\PycharmPr
Pregnancies is with coefficient of 0.4086468712956544
Glucose is with coefficient of 1.107111971571363
BloodPressure is with coefficient of -0.2508680388517632
SkinThickness is with coefficient of 0.00905045508333392
Insulin is with coefficient of -0.13083627394872085
BMI is with coefficient of 0.6963087173103848
DiabetesPedigreeFunction is with coefficient of 0.3088372414180195
Age is with coefficient of 0.1764978217442397

Dimensionality Reduction



01

Feature Selection

Original features are maintained

Keep most important features

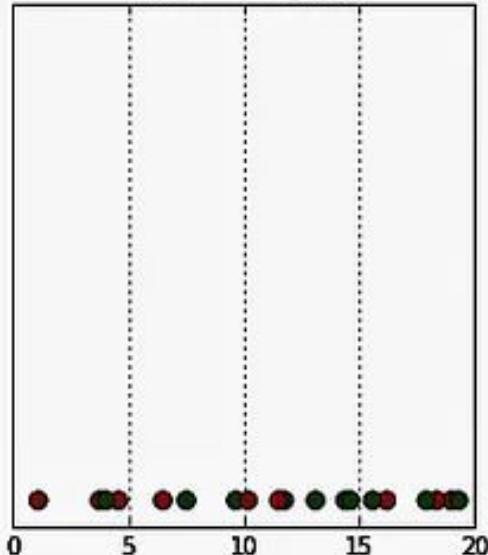
02

Feature Extraction

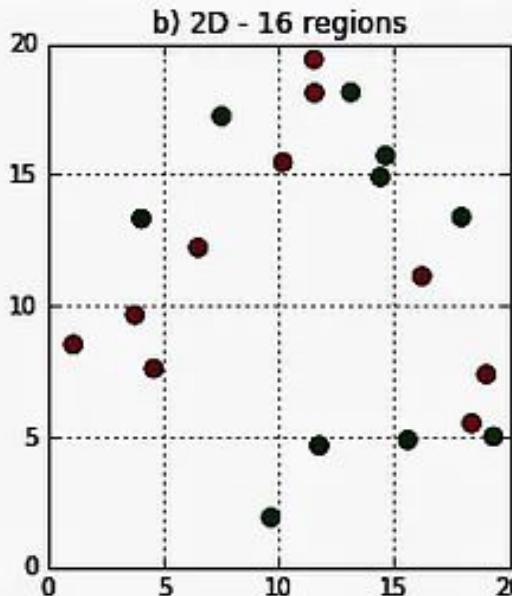
Features are transformed to a new space

Curse of Dimensionality

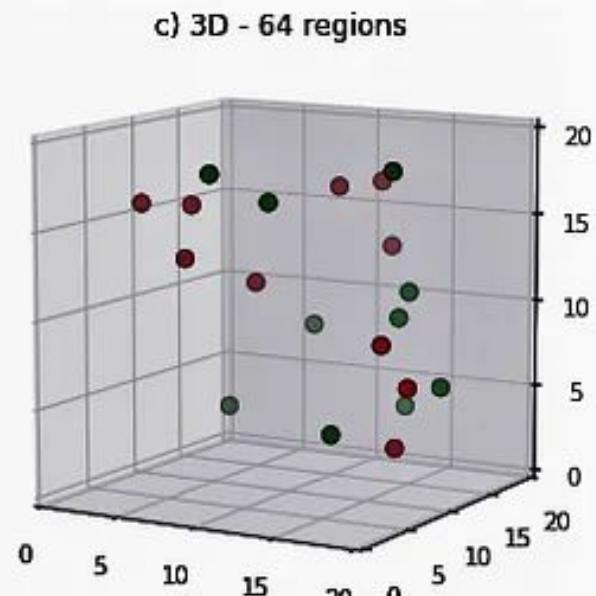
a) 1D - 4 regions



b) 2D - 16 regions



c) 3D - 64 regions



Feature Selection

Feature Selection

Full Feature Set



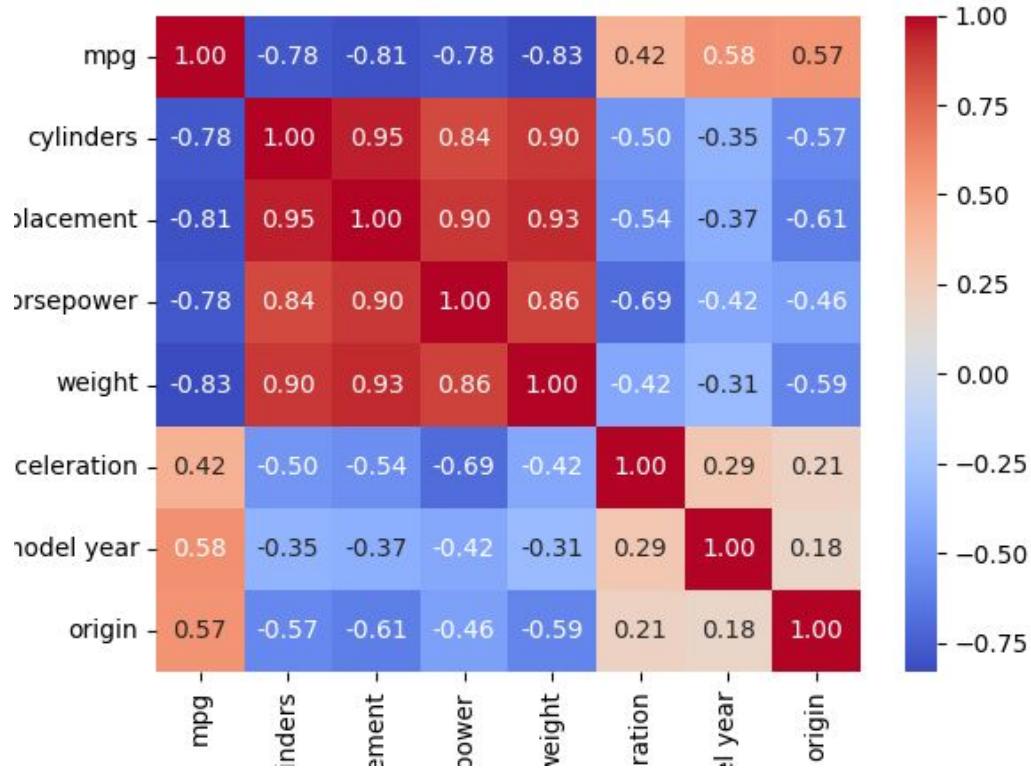
Identify Useful Features



Selected Feature Set



Feature Selection - Correlation Coefficient



Feature Selection - Variance Threshold

```
import pandas as pd
from sklearn.feature_selection import VarianceThreshold

data = pd.read_csv("diabetes.csv", usecols=[i for i in range(8)])
variance_threshold = VarianceThreshold(threshold=1)
variance_threshold.fit(data)
print(variance_threshold.variances_)
print(variance_threshold.get_support())
```

fuel_viz ✘

```
C:\Users\Viet\anaconda3\envs\basic\python.exe C:\Users\Viet\Pychari
[1.13392724e+01 1.02091726e+03 3.74159449e+02 2.54141900e+02
 1.32638869e+04 6.20790465e+01 1.09635697e-01 1.38122964e+02]
[ True  True  True  True  True  True False  True]
```

Feature Selection - Lasso (L1)

```
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline

data = pd.read_csv("diabetes.csv")
x = data.drop("Outcome", axis=1)
y = data["Outcome"]
cls = Pipeline(steps=[
    ('scaler', StandardScaler()),
    ('model', LogisticRegression(penalty="l1", solver="liblinear"))
])
cls.fit(x, y)
for i, j in zip(x.columns, cls["model"].coef_[0]):
    print("{} is with coefficient of {}".format(i, j))
```

fuel_viz ×
C:\Users\Viet\anaconda3\envs\basic\python.exe C:\Users\Viet\PycharmPr
Pregnancies is with coefficient of 0.4047757715484604
Glucose is with coefficient of 1.1020534150332097
BloodPressure is with coefficient of -0.23905878835657515
SkinThickness is with coefficient of 0.0
Insulin is with coefficient of -0.12014123606887196
BMI is with coefficient of 0.6883338771775394
DiabetesPedigreeFunction is with coefficient of 0.30157785094254597
Age is with coefficient of 0.16766878675897076

```
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline

data = pd.read_csv("diabetes.csv")
x = data.drop("Outcome", axis=1)
y = data["Outcome"]
cls = Pipeline(steps=[
    ('scaler', StandardScaler()),
    ('model', LogisticRegression(penalty="l2", solver="lbfgs"))
])
cls.fit(x, y)
for i, j in zip(x.columns, cls["model"].coef_[0]):
    print("{} is with coefficient of {}".format(i, j))
```

fuel_viz ×
C:\Users\Viet\anaconda3\envs\basic\python.exe C:\Users\Viet\PycharmPr
Pregnancies is with coefficient of 0.4086468712956544
Glucose is with coefficient of 1.107111971571363
BloodPressure is with coefficient of -0.2508680388517632
SkinThickness is with coefficient of 0.00905045508333392
Insulin is with coefficient of -0.13083627394872085
BMI is with coefficient of 0.6963087173103848
DiabetesPedigreeFunction is with coefficient of 0.3088372414180195
Age is with coefficient of 0.1764978217442397

Feature Selection - RandomForest

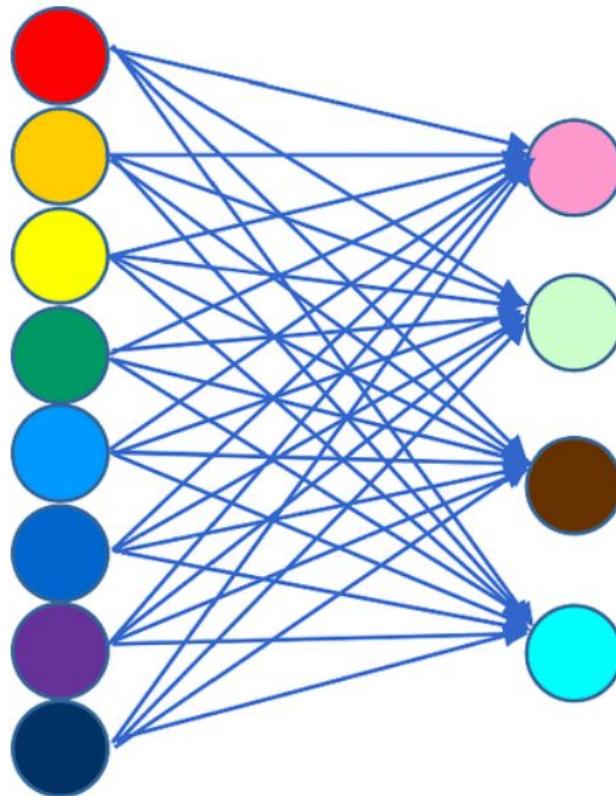
```
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline

data = pd.read_csv("diabetes.csv")
x = data.drop("Outcome", axis=1)
y = data["Outcome"]
cls = Pipeline(steps=[
    ('scaler', StandardScaler()),
    ('model', RandomForestClassifier())
])
cls.fit(x, y)
print(cls["model"].feature_importances_)
```

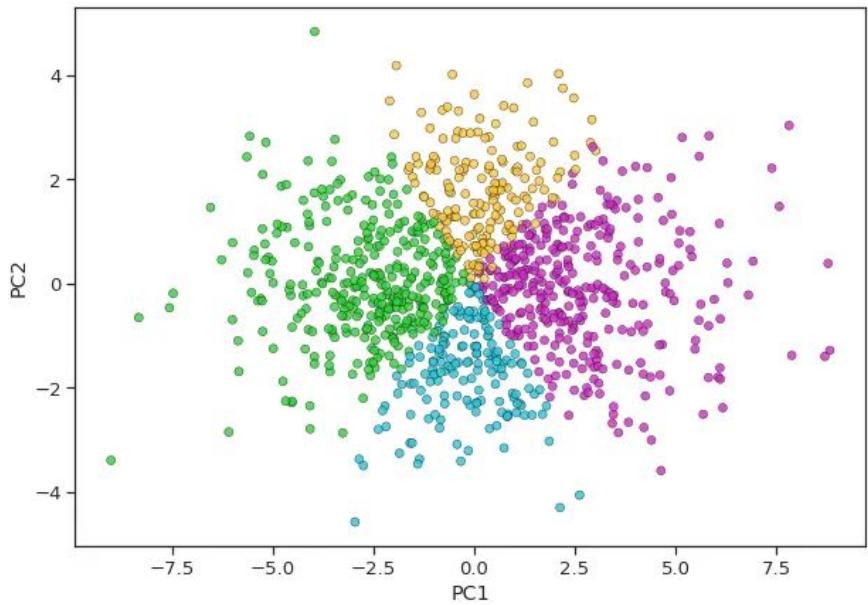
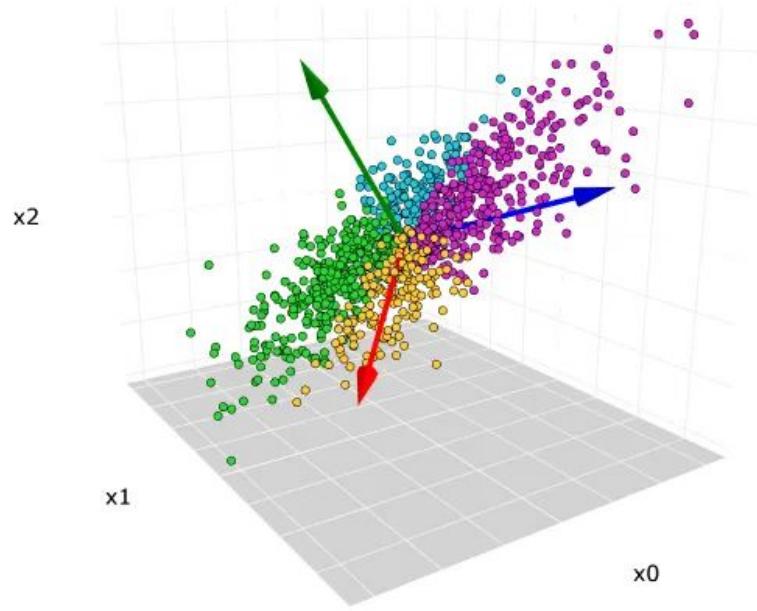
fuel_viz ×

```
C:\Users\Viet\anaconda3\envs\basic\python.exe C:/Users/Viet/Pycharm
[0.08485069 0.26544363 0.09088871 0.07260613 0.07047188 0.15482689
 0.12826565 0.13264643]
```

Feature Extraction

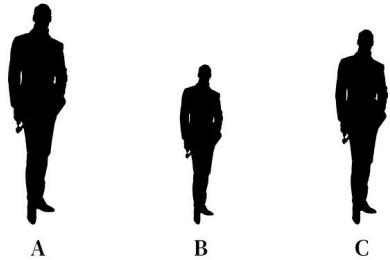


Feature Extraction - Principle Component Analyst

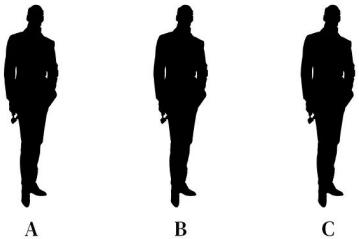


Variance in data

$$\begin{aligned}\text{Var}(X) &= \mathbb{E}[(X - \mathbb{E}[X])^2] \\ &= \mathbb{E}[X^2 - 2X\mathbb{E}[X] + \mathbb{E}[X]^2] \\ &= \mathbb{E}[X^2] - 2\mathbb{E}[X]\mathbb{E}[X] + \mathbb{E}[X]^2 \\ &= \mathbb{E}[X^2] - \mathbb{E}[X]^2\end{aligned}$$



Person	Height (cm)
Alex	145
Ben	160
Chris	185



Person	Height (cm)
Daniel	172
Elsa	173
Fernandez	171

Covariance vs Correlation

$$\begin{aligned}\text{Var}(X) &= E[(X - E[X])^2] \\&= E[X^2 - 2XE[X] + E[X]^2] \\&= E[X^2] - 2E[X]E[X] + E[X]^2 \\&= E[X^2] - E[X]^2\end{aligned}$$

$$\begin{aligned}\text{cov}(X, Y) &= E[(X - E[X])(Y - E[Y])] \\&= E[XY - XE[Y] - E[X]Y + E[X]E[Y]] \\&= E[XY] - E[X]E[Y] - E[X]E[Y] + E[X]E[Y] \\&= E[XY] - E[X]E[Y],\end{aligned}$$

$$\text{Corr}(X, Y) = \frac{\text{Cov}(X, Y)}{\sigma_x \sigma_y} \quad \left. \vphantom{\frac{\text{Cov}(X, Y)}{\sigma_x \sigma_y}} \right\} \text{Covariance normalized by Standard Deviation}$$

Correlation between X and Y

Standard deviation of X

Standard deviation of Y

Standard Deviation = $\sqrt{\text{Variance}}$

Covariance vs Correlation

Covariance	Correlation
Indicates the direction of the linear relationship between variables	Indicates both the strength and direction of the linear relationship between two variables
Covariance values are not standard	Correlation values are standardized
Positive number being positive relationship and negative number being negative relationship	1 being strong positive correlation, -1 being strong negative correlation
Value between positive infinity to negative infinity	Value is strictly between -1 to 1

PCA: step 1 - Standardize the dataset

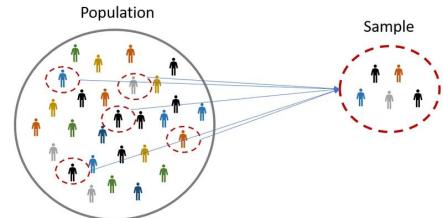
f1	f2	f3	f4
1	2	3	4
5	5	6	7
1	4	2	3
5	3	2	1
8	1	2	2

$$x_{new} = \frac{x - \mu}{\sigma}$$

f1	f2	f3	f4
-1	-0.63246	0	0.26062
0.33333	1.26491	1.73205	1.56374
-1	0.63246	-0.57735	-0.17375
0.33333	0	-0.57735	-1.04249
1.33333	-1.26491	-0.57735	-0.60812

μ	=	f1	f2	f3	f4
0.33333		4	3	3	3.4
1.33333		3	1.58114	1.73205	2.30217

PCA: step 2 - Calculate the covariance matrix



f1	f2	f3	f4
-1	-0.63246	0	0.26062
0.33333	1.26491	1.73205	1.56374
-1	0.63246	-0.57735	-0.17375
0.33333	0	-0.57735	-1.04249
1.33333	-1.26491	-0.57735	-0.60812

For Population

$$\text{Cov}(x,y) = \frac{\sum (x_i - \bar{x}) * (y_i - \bar{y})}{N}$$

For Sample

$$\text{Cov}(x,y) = \frac{\sum (x_i - \bar{x}) * (y_i - \bar{y})}{(N - 1)}$$



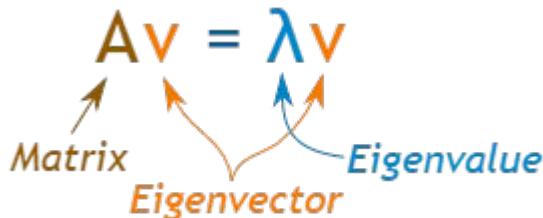
	f1	f2	f3	f4
f1	0.8	-0.25298	0.03849	-0.14479
f2	-0.25298	0.8	0.51121	0.4945
f3	0.03849	0.51121	0.8	0.75236
f4	-0.14479	0.4945	0.75236	0.8

	f1	f2	f3	f4
f1	var(f1)	cov(f1,f2)	cov(f1,f3)	cov(f1,f4)
f2	cov(f2,f1)	var(f2)	cov(f2,f3)	cov(f2,f4)
f3	cov(f3,f1)	cov(f3,f2)	var(f3)	cov(f3,f4)
f4	cov(f4,f1)	cov(f4,f2)	cov(f4,f3)	var(f4)



PCA: step 3 - Calculate eigenvector and eigenvalue

⚠ **Definition.** Let A be an $n \times n$ matrix.



1. An *eigenvector* of A is a *nonzero* vector v in \mathbb{R}^n such that $Av = \lambda v$, for some scalar λ .
2. An *eigenvalue* of A is a scalar λ such that the equation $Av = \lambda v$ has a *nontrivial* solution.

If $Av = \lambda v$ for $v \neq 0$, we say that λ is the *eigenvalue for v*, and that v is an *eigenvector for λ* .

$$A = \begin{bmatrix} 1 & 4 \\ 3 & 2 \end{bmatrix}$$
$$\det(A - \lambda I) = 0$$
$$\det\left(\begin{bmatrix} 1 & 4 \\ 3 & 2 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right) = 0$$
$$\det\left(\begin{bmatrix} 1-\lambda & 4 \\ 3 & 2-\lambda \end{bmatrix}\right) = 0$$

$$\det \begin{bmatrix} a & b \\ c & d \end{bmatrix} = ad - bc$$
$$(1-\lambda)(2-\lambda) - 12 = 0$$

$$\lambda^2 - 3\lambda - 10 = 0$$

$$(\lambda - 5)(\lambda + 2) = 0$$

$$\lambda = 5, -2$$

PCA: step 3 - Calculate eigenvector and eigenvalue

	f1	f2	f3	f4
f1	0.8	-0.25298	0.03849	-0.14479
f2	-0.25298	0.8	0.51121	0.4945
f3	0.03849	0.51121	0.8	0.75236
f4	-0.14479	0.4945	0.75236	0.8

	f1	f2	f3	f4
f1	0.8 - λ	-0.25298	0.03849	-0.14479
f2	-0.25298	0.8 - λ	0.51121	0.4945
f3	0.03849	0.51121	0.8 - λ	0.75236
f4	-0.14479	0.4945	0.75236	0.8 - λ

λ	0.393	1.065	2.515	0.025
-----------	-------	-------	-------	-------

v	-0.307	-0.917	0.161	0.196
	-0.817	0.206	-0.524	0.120
	0.188	-0.320	-0.585	-0.720
	0.449	-0.115	-0.596	0.654

PCA: step 4 - Sort eigenvalues & corresponding eigenvectors

λ	0.393	1.065	2.515	0.025

λ	2.515	1.065	0.393	0.025

v	-0.307	-0.917	0.161	0.196
v	-0.817	0.206	-0.524	0.120
v	0.188	-0.320	-0.585	-0.720
v	0.449	-0.115	-0.596	0.654

v	0.161	-0.917	-0.307	0.196
v	-0.524	0.206	-0.817	0.120
v	-0.585	-0.320	0.188	-0.720
v	-0.596	-0.115	0.449	0.654

PCA: step 5 - Pick k eigenvalues and form a matrix

$k=2$

The diagram illustrates the fifth step of PCA, where we select the top $k=2$ eigenvalues and their corresponding eigenvectors to form a matrix. A red arrow points down from the value 1.065 in the eigenvalue list to the first row of the eigenvector matrix. Another red arrow points down from the value 0.393 to the second row.

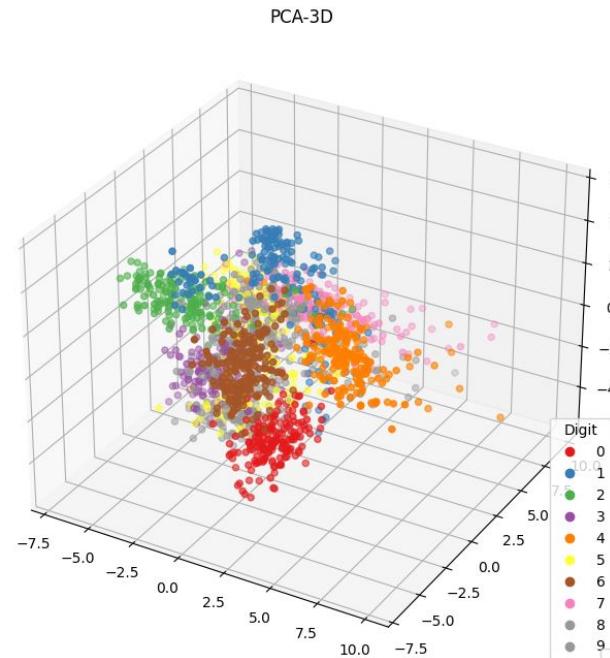
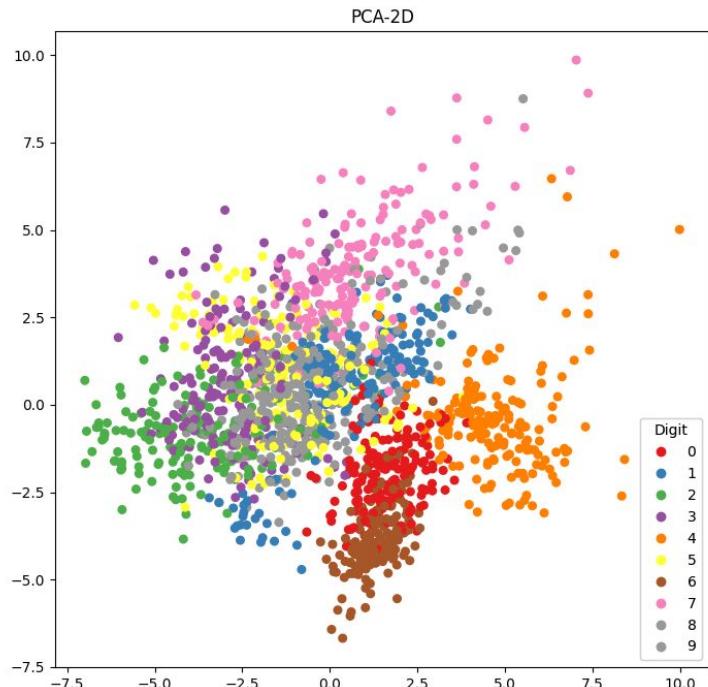
λ	2.515	1.065	0.393	0.025
v	0.161	-0.917		
	-0.524	0.206		
	-0.585	-0.320		
	-0.596	-0.115		

PCA: step 6 - Transform feature matrix

$$\begin{array}{c} \text{Nx4 matrix} \\ \times \\ \left[\begin{array}{cc} 0.161 & -0.917 \\ -0.524 & 0.206 \\ -0.585 & -0.320 \\ -0.596 & -0.115 \end{array} \right] \\ = \\ \text{Nx2 matrix} \end{array}$$

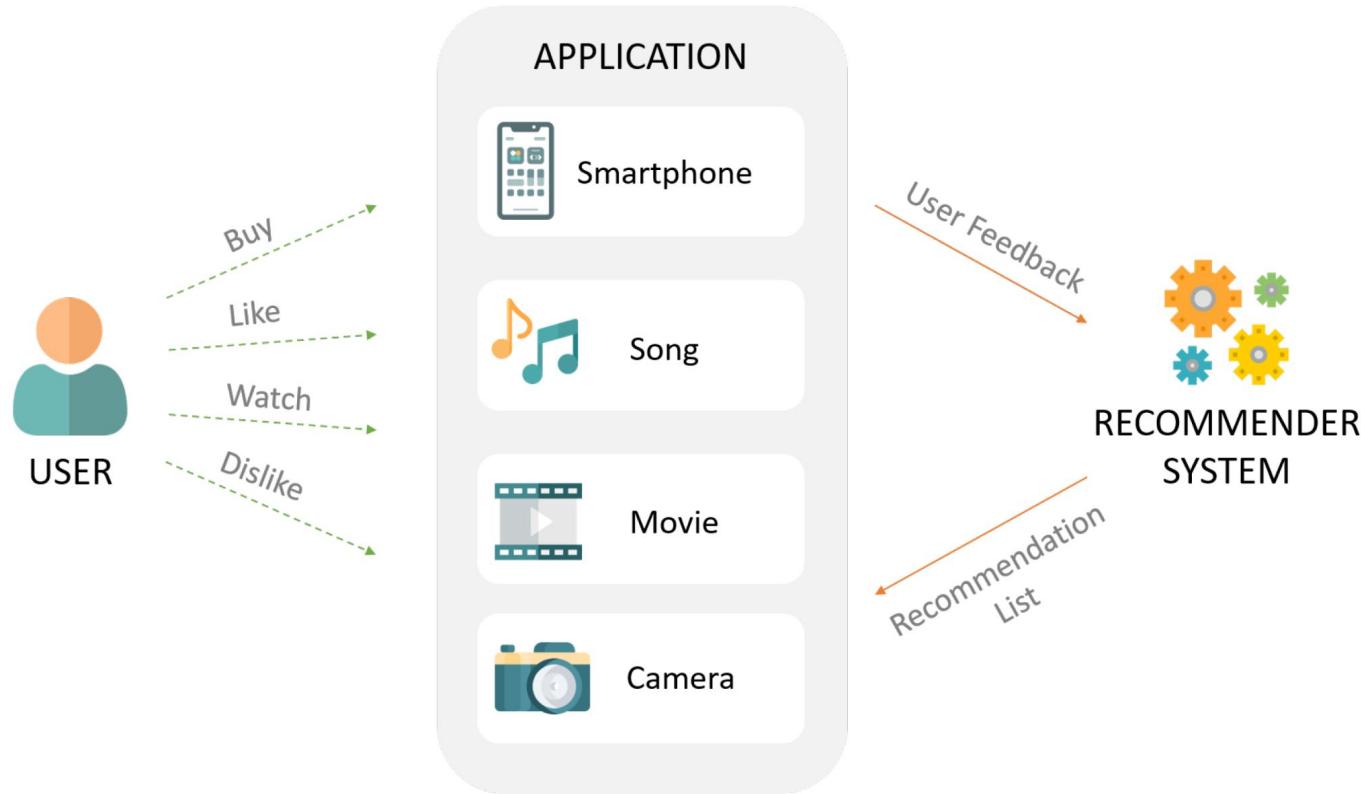
PCA visualization

MNIST Visualization

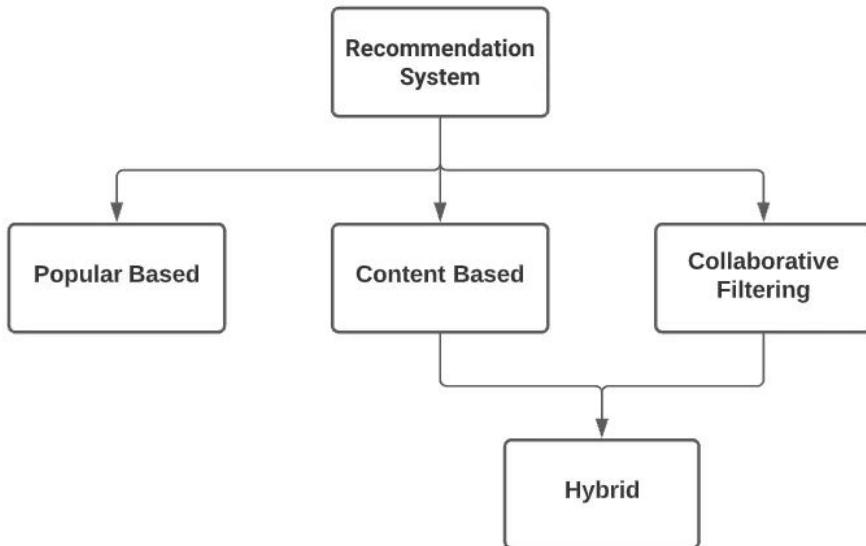


Recommendation systems

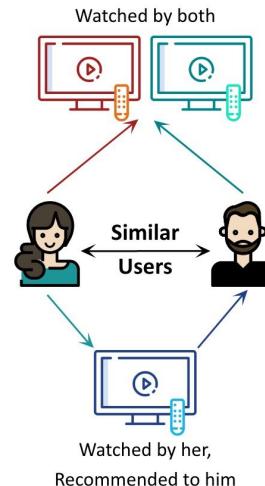
What is recommendation systems



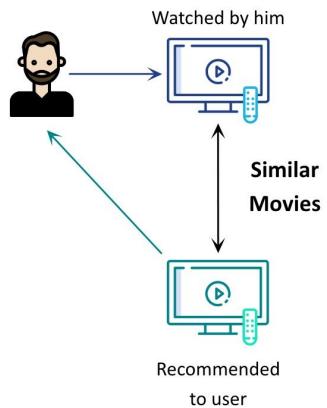
Types of recommendation systems



Collaborative Filtering



Content-Based Filtering



Popularity-based recommendation systems

Trending Now



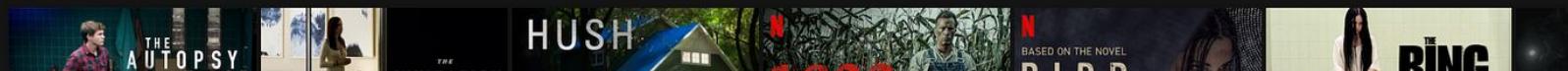
Top 10 Movies in the U.S. Today



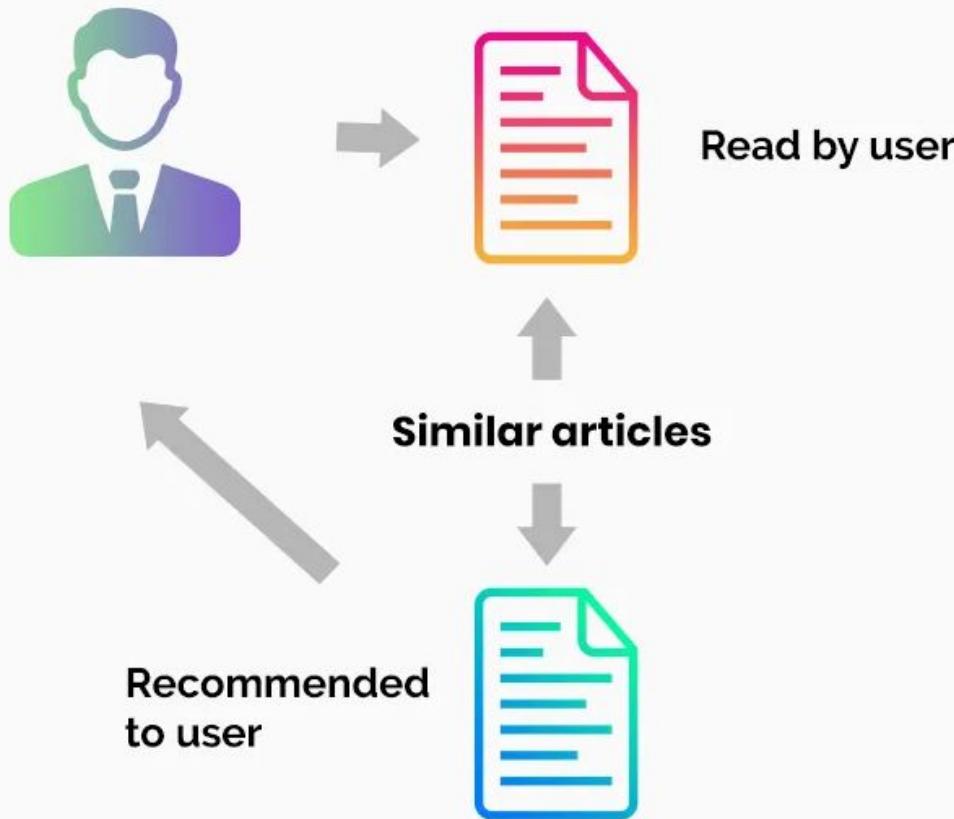
New Releases



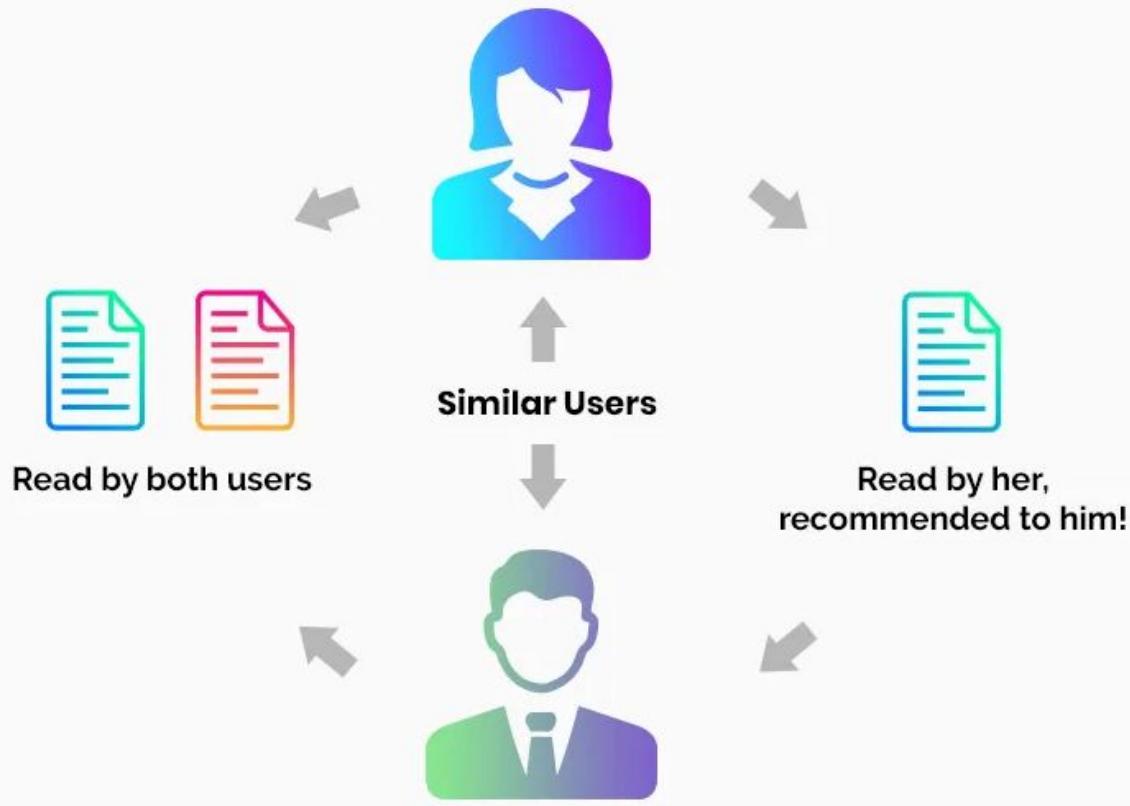
Scary Movies



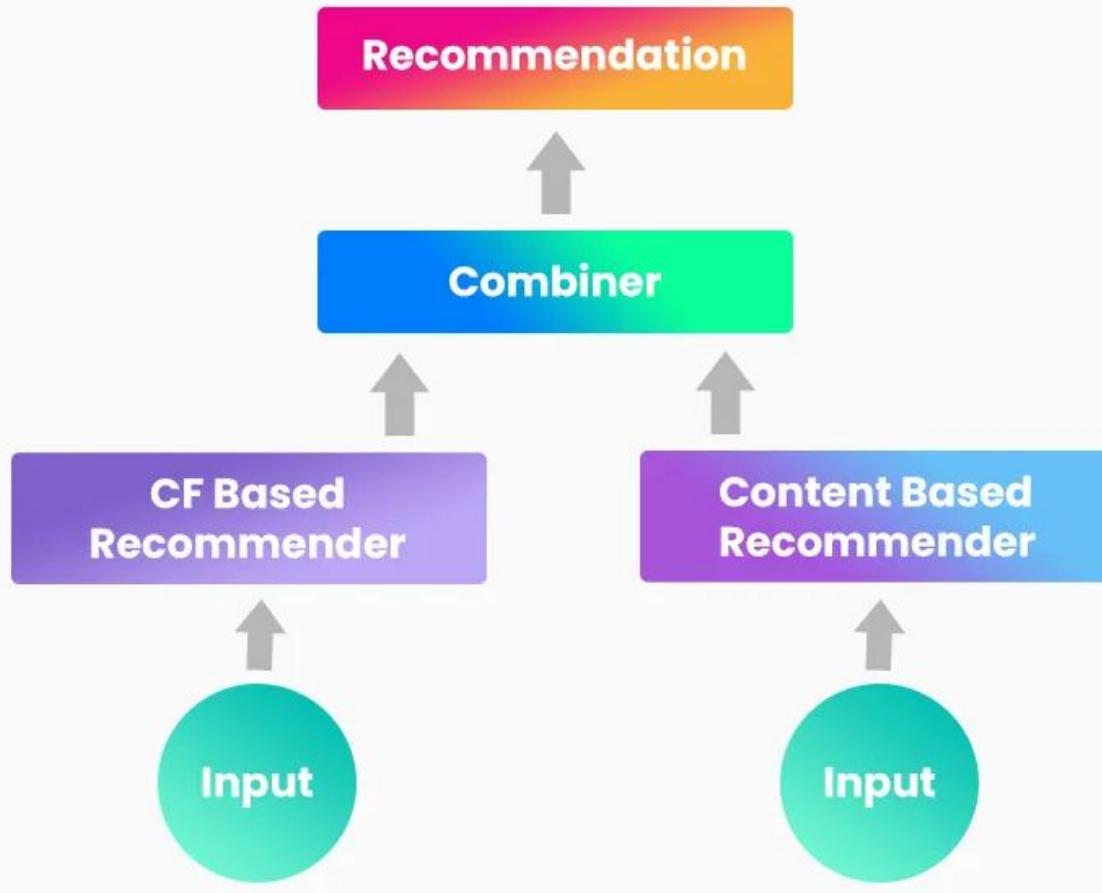
Content-based filtering



Collaborative filtering



Hybrid Recommendations



Utility matrix

	Item 1	Item 2	Item 3	Item 4	Item 5
User 1	●			●	
User 2		●			●
User 3	●		●		
User 4			●		