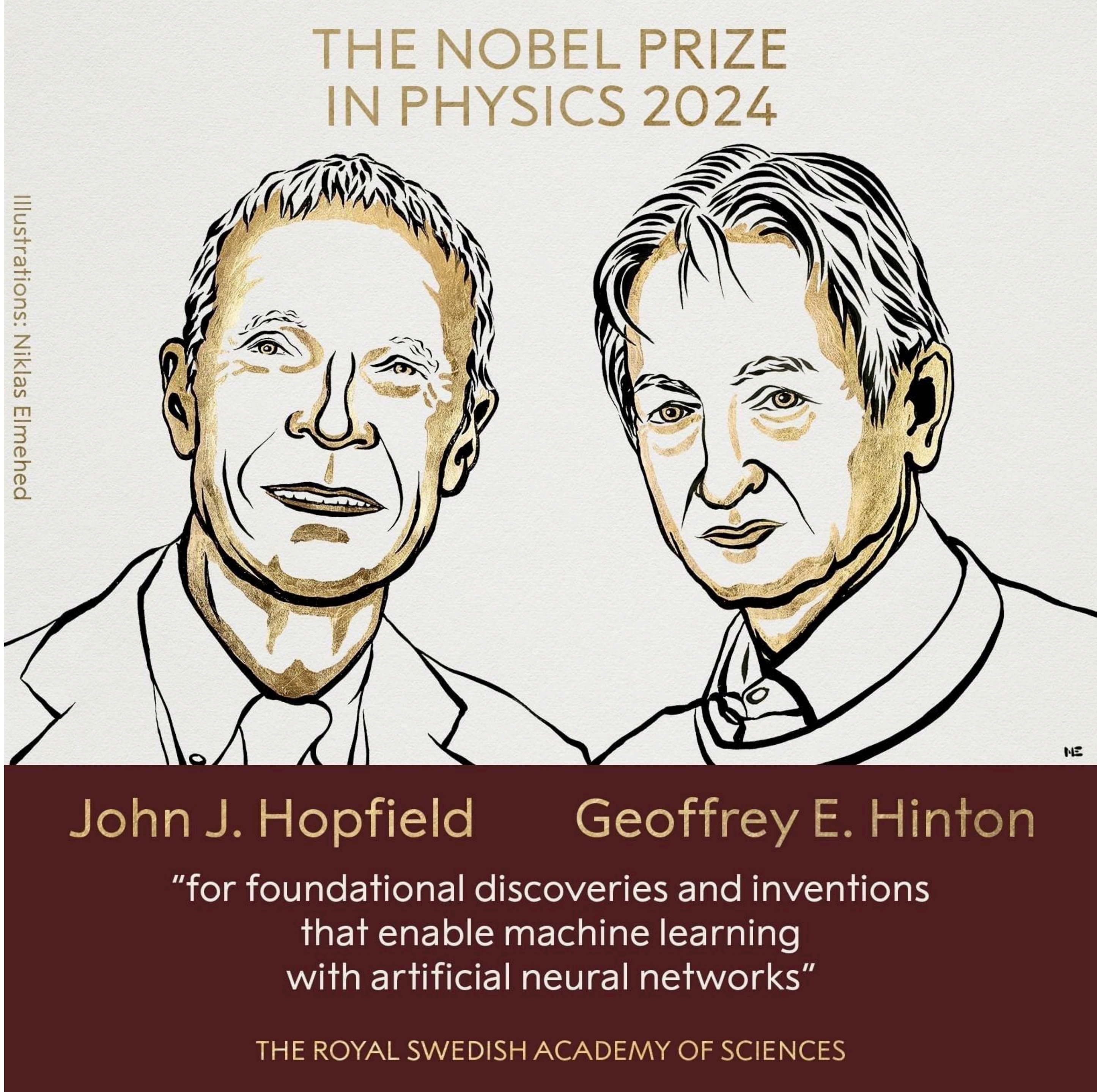
Classical Machine Learning: Classification and Regression (IV)

- Learn how to fine-tune your machine learning model.
- Learn how to choose the best machine learning model.
- Learn how to deal with class imbalance problems (with some useful performance metrics)

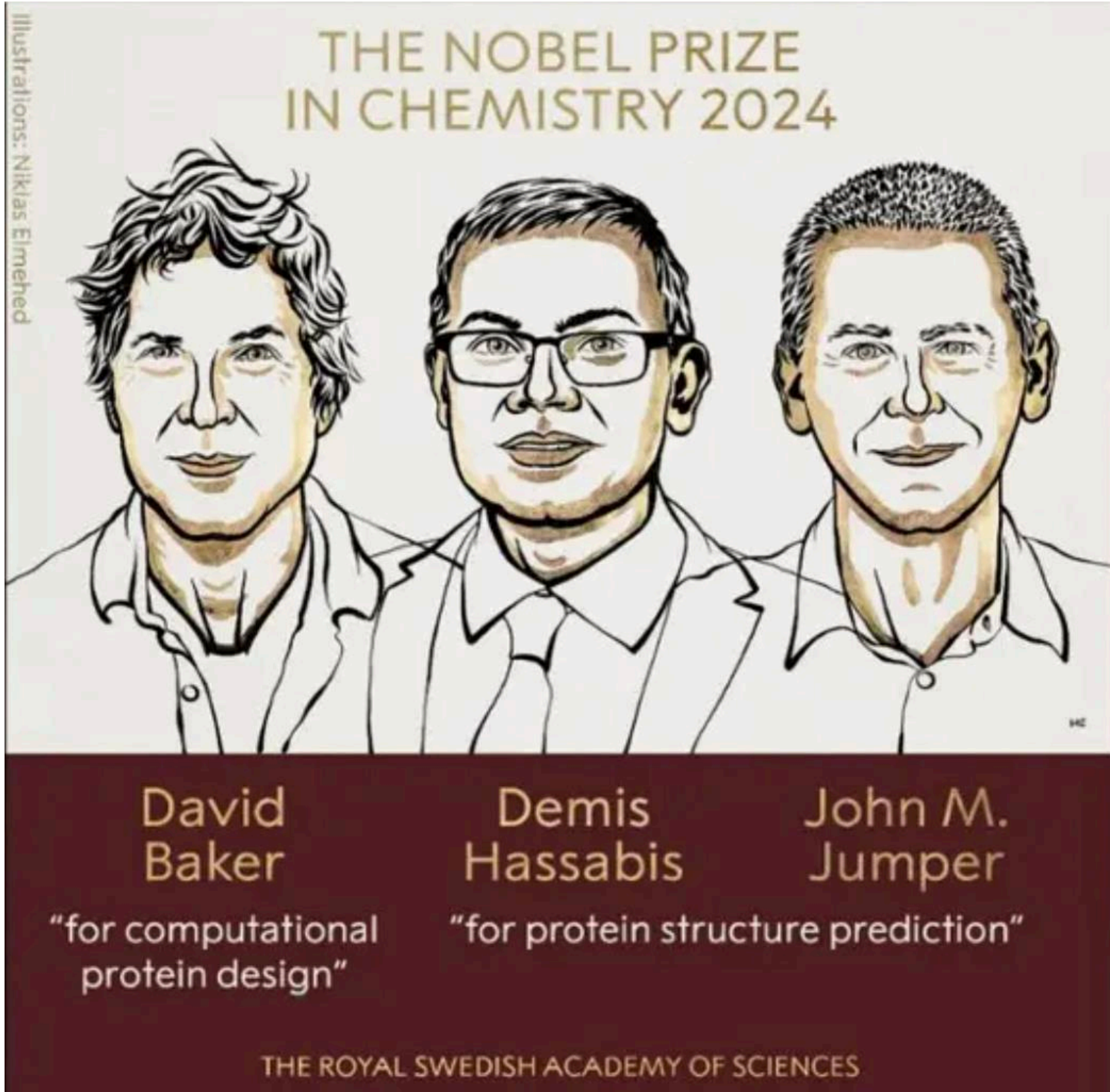
汝為Ai人，不可不知Ai事

2024.10.08


THE NOBEL PRIZE
IN PHYSICS 2024
Illustrations: Niklas Elmehed
NE
John J. Hopfield
Geoffrey E. Hinton
“for foundational discoveries and inventions that enable machine learning with artificial neural networks”
THE ROYAL SWEDISH ACADEMY OF SCIENCES

汝為Ai人，不可不知Ai事

2024.10.09



THE NOBEL PRIZE IN CHEMISTRY 2024

David Baker “for computational protein design”

Demis Hassabis and John M. Jumper “for protein structure prediction”

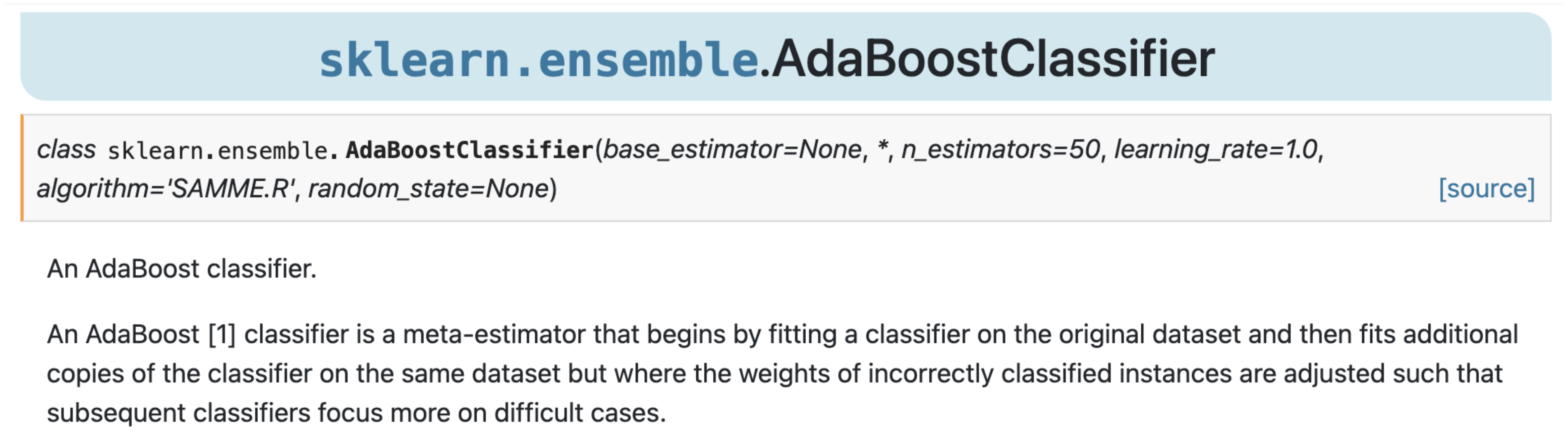
THE ROYAL SWEDISH ACADEMY OF SCIENCES

UNGL.
ETENSKAPS
KADEMIEN
#NobelPrize
THE NOBEL PRIZE

Algorithm Tuning: AdaBoost, Gradient Boost

Algorithm Tuning

- Algorithm tuning mainly involves tuning **hyperparameters**.
- Hyperparameters are parameters that are not directly learnt within estimators (algorithms).
- In scikit-learn they are passed as arguments to the constructor of the estimator classes.
- Two generic approaches to parameter search are provided in scikit-learn: for given values, **GridSearchCV** exhaustively considers all parameter combinations, while **RandomizedSearchCV** can sample a given number of candidates from a parameter space with a specified distribution.

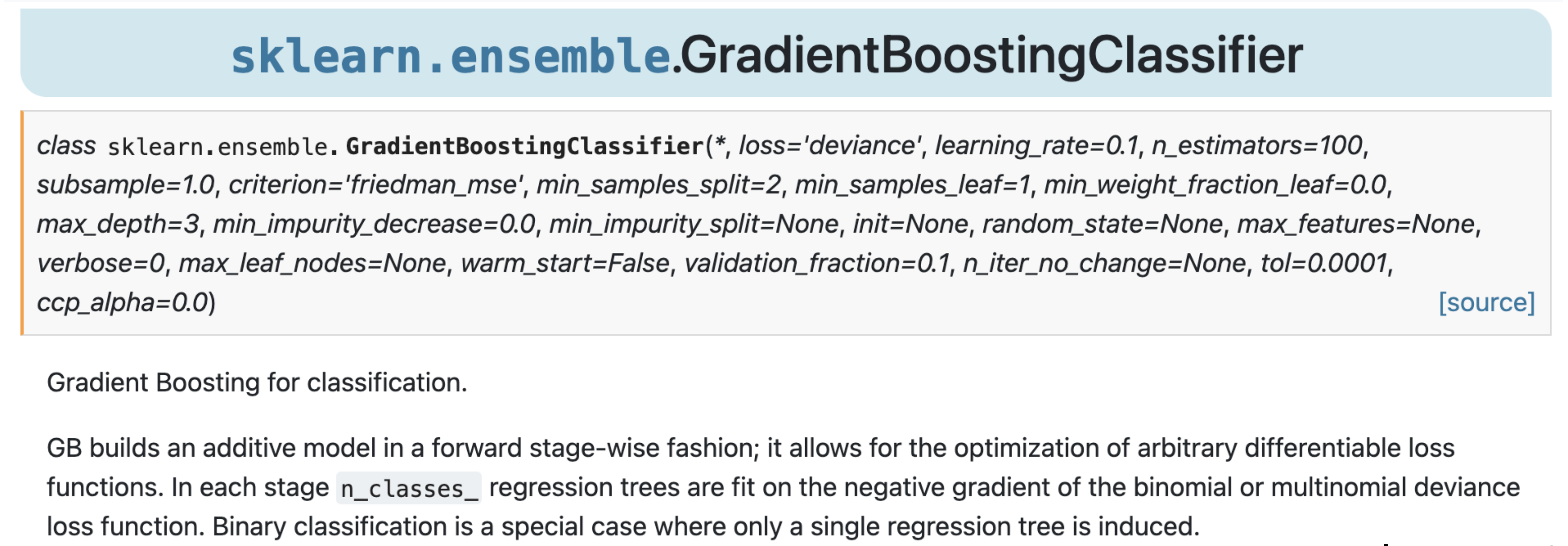


sklearn.ensemble.AdaBoostClassifier

class `sklearn.ensemble.`**`AdaBoostClassifier`**(*base_estimator=None*, *, *n_estimators=50*, *learning_rate=1.0*, *algorithm='SAMME.R'*, *random_state=None*) [source]

An AdaBoost classifier.

An AdaBoost [1] classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases.



sklearn.ensemble.GradientBoostingClassifier

class `sklearn.ensemble.`**`GradientBoostingClassifier`**(*, *loss='deviance'*, *learning_rate=0.1*, *n_estimators=100*, *subsample=1.0*, *criterion='friedman_mse'*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_depth=3*, *min_impurity_decrease=0.0*, *min_impurity_split=None*, *init=None*, *random_state=None*, *max_features=None*, *verbose=0*, *max_leaf_nodes=None*, *warm_start=False*, *validation_fraction=0.1*, *n_iter_no_change=None*, *tol=0.0001*, *ccp_alpha=0.0*) [source]

Gradient Boosting for classification.

GB builds an additive model in a forward stage-wise fashion; it allows for the optimization of arbitrary differentiable loss functions. In each stage `n_classes_` regression trees are fit on the negative gradient of the binomial or multinomial deviance loss function. Binary classification is a special case where only a single regression tree is induced.

Algo_Tuning.ipynb

Class imbalance problems and some alternative performance metrics

Class Imbalance Problem

- Lots of classification problems where the classes are skewed (more records from one class than another)
  - Credit card fraud
  - Intrusion detection
  - Defective products in manufacturing assembly line
  - COVID-19 test results on a random sample

- **Key Challenge**:
  - Evaluation measures such as accuracy are not well-suited for imbalanced class

Class Imbalance Techniques

- Choose better performance metrics
- Balance skewed classes
- Cost-sensitive learning

Confusion Matrix

- Confusion Matrix:

	PREDICTED CLASS		
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a	b
	Class=No	c	d

a: TP (true positive)

b: FN (false negative)

c: FP (false positive)

d: TN (true negative)

Accuracy

	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	a (TP)	b (FN)
	Class=No	c (FP)	d (TN)

- Most widely-used metric:

$$\text{Accuracy} = \frac{a+d}{a+b+c+d} = \frac{TP+TN}{TP+TN+FP+FN}$$

Problem with Accuracy

- Consider a 2-class problem
  - Number of Class NO examples = 990
  - Number of Class YES examples = 10
- If a model predicts everything to be class NO, accuracy is 990/1000 = 99 %
  - This is misleading because this trivial model does not detect any class YES example
  - Detecting the rare class is usually more interesting (e.g., frauds, intrusions, defects, etc)

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	0	10
	Class=No	0	990

Alternative Measures

	PREDICTED CLASS		
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a	b
	Class=No	c	d

$$\text{Precision (p)} = \frac{a}{a+c}$$

$$\text{Recall (r)} = \frac{a}{a+b}$$

$$\text{F - measure (F)} = \frac{2rp}{r+p} = \frac{2a}{2a+b+c}$$

Alternative Measures

	PREDICTED CLASS		
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	10	0
	Class=No	10	980

$$\text{Precision (p)} = \frac{10}{10+10} = 0.5$$

$$\text{Recall (r)} = \frac{10}{10+0} = 1$$

$$\text{F - measure (F)} = \frac{2*1*0.5}{1+0.5} = 0.62$$

$$\text{Accuracy} = \frac{990}{1000} = 0.99$$

	PREDICTED CLASS		
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	1	9
	Class=No	0	990

$$\text{Precision (p)} = \frac{1}{1+0} = 1$$

$$\text{Recall (r)} = \frac{1}{1+9} = 0.1$$

$$\text{F - measure (F)} = \frac{2*0.1*1}{1+0.1} = 0.18$$

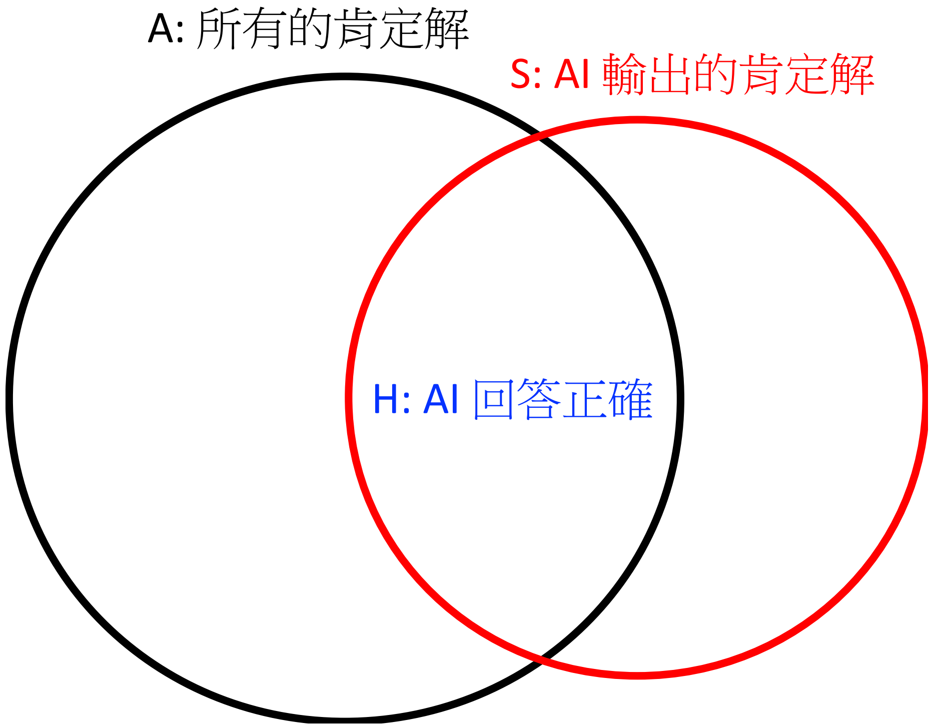
$$\text{Accuracy} = \frac{991}{1000} = 0.991$$

Recall (查全率) and Precision (精確率)

	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	a	b
	Class=No	c	d

$$\text{Recall} = \frac{a}{a+b}$$

$$\text{Precision} = \frac{a}{a+c}$$



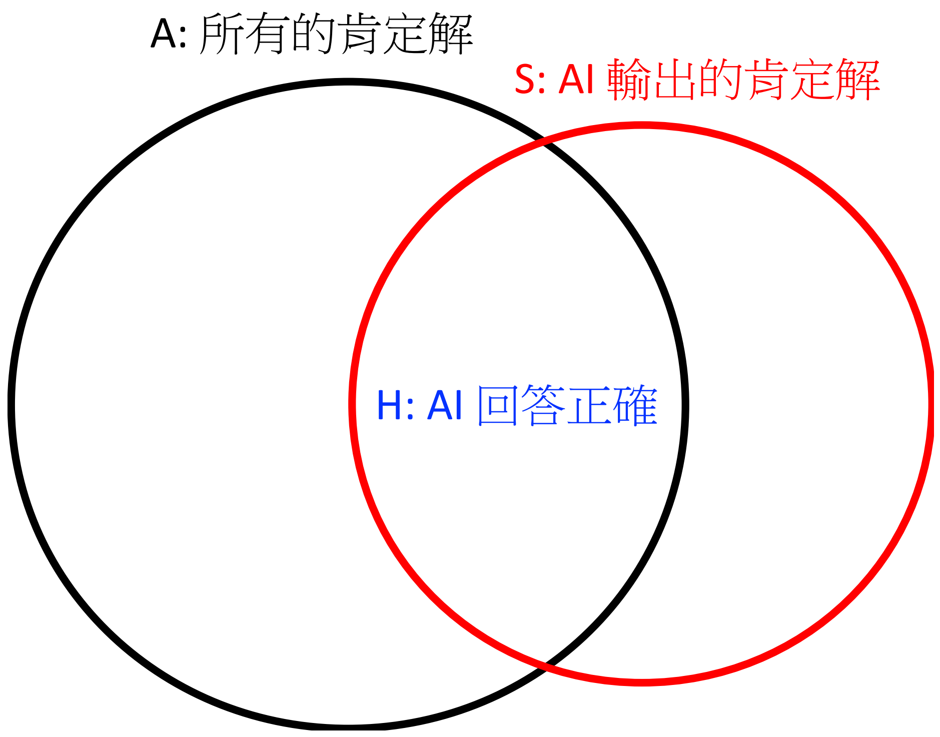
$$\text{Recall (查全率)} = \frac{H}{A}$$

$$\text{Precision (精確率)} = \frac{H}{S}$$

Fun Time:

要重視查全率還是精確率?

「寧可錯殺，不可錯放」重視的是 (1)查全率Recall (2) 精確率Precision



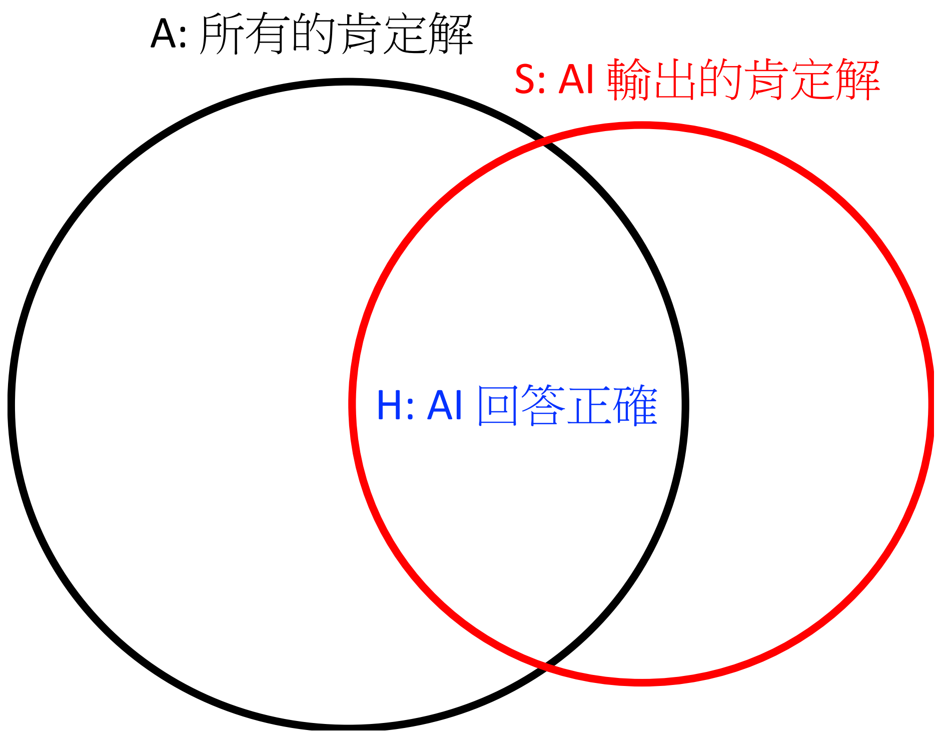
$\text{Recall (查全率)} = \frac{H}{A}$

$\text{Precision (精確率)} = \frac{H}{S}$

Fun Time:

要重視查全率還是精確率?

開發人臉辨識系統我們應該要重視查全率 Recall 還是精確率 Precision? (1)查全率 Recall (2) 精確率Precision



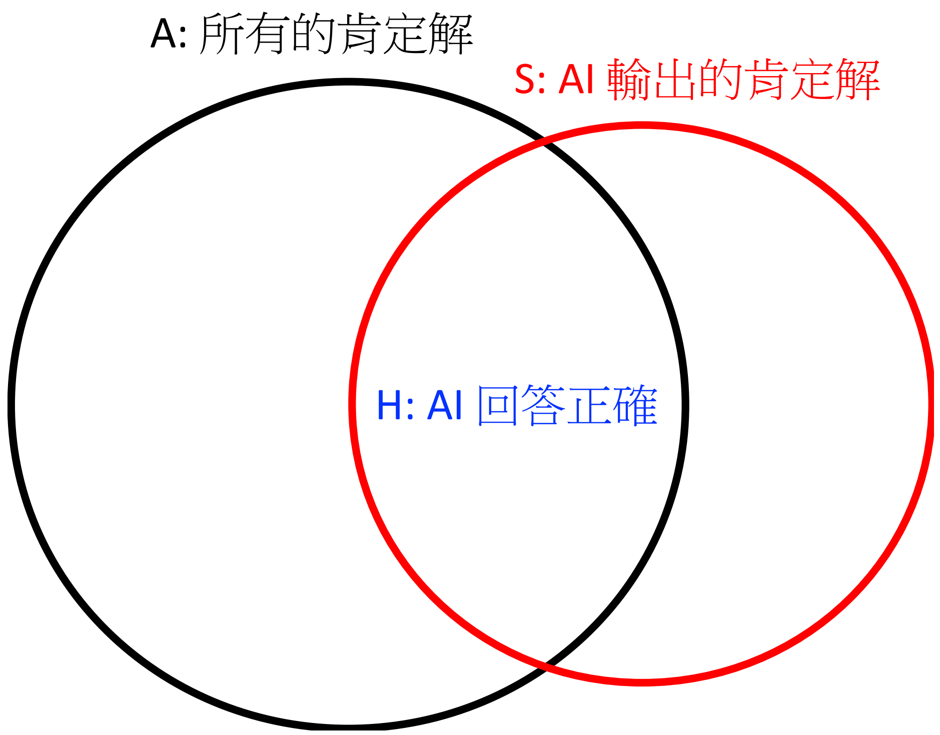
$$\text{Recall (查全率)} = \frac{H}{A}$$

$$\text{Precision (精確率)} = \frac{H}{S}$$

Fun Time:

要重視查全率還是精確率?

開發癌症早期檢測系統我們應該要重視查全率 Recall 還是精確率 Precision? (1)查全率 Recall (2) 精確率Precision



$$\text{Recall (查全率)} = \frac{H}{A}$$

$$\text{Precision (精確率)} = \frac{H}{S}$$

Measures of Classification Performance

		PREDICTED CLASS	
		Yes	No
ACTUAL CLASS	Yes	TP	FN
	No	FP	TN

α is a Type I error or a false positive (FP).

β is a Type II error or a false negative (FN).

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN}$$

$$ErrorRate = 1 - accuracy$$

$$Precision = Positive\ Predictive\ Value = \frac{TP}{TP + FP}$$

$$Recall = Sensitivity = TP\ Rate = \frac{TP}{TP + FN}$$

$$Specificity = TN\ Rate = \frac{TN}{TN + FP}$$

$$FP\ Rate = \alpha = \frac{FP}{TN + FP} = 1 - specificity$$

$$FN\ Rate = \beta = \frac{FN}{FN + TP} = 1 - sensitivity$$

$$Power = sensitivity = 1 - \beta$$

ROC (Receiver Operating Characteristic)

- A graphical approach for displaying trade-off between detection rate and false alarm rate
- Developed in 1950s for signal detection theory to analyze noisy signals
- ROC curve plots TPR against FPR
  - Performance of a model represented as a point in an ROC curve

	PREDICTED CLASS		
		Yes	No
ACTUAL CLASS	Yes	TP	FN
	No	FP	TN

$$TPR = recall = \frac{TP}{TP + FN}$$

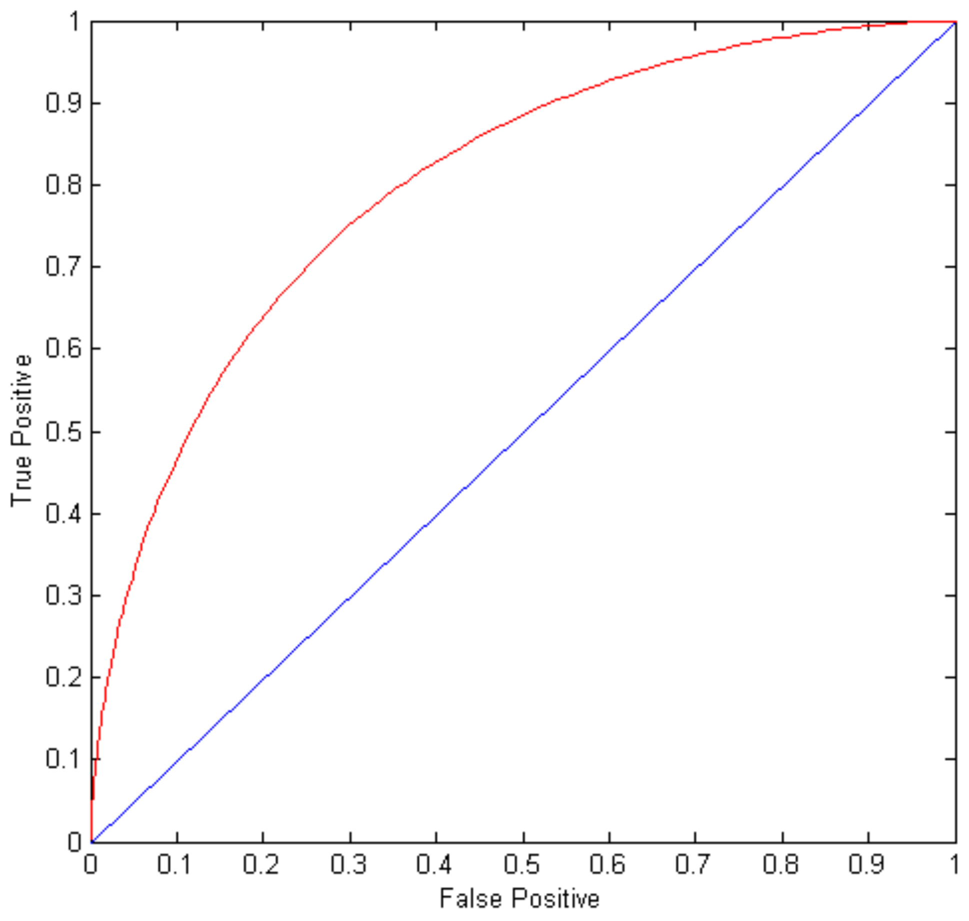
$$FPR = \frac{FP}{TN + FP}$$

ROC Curve

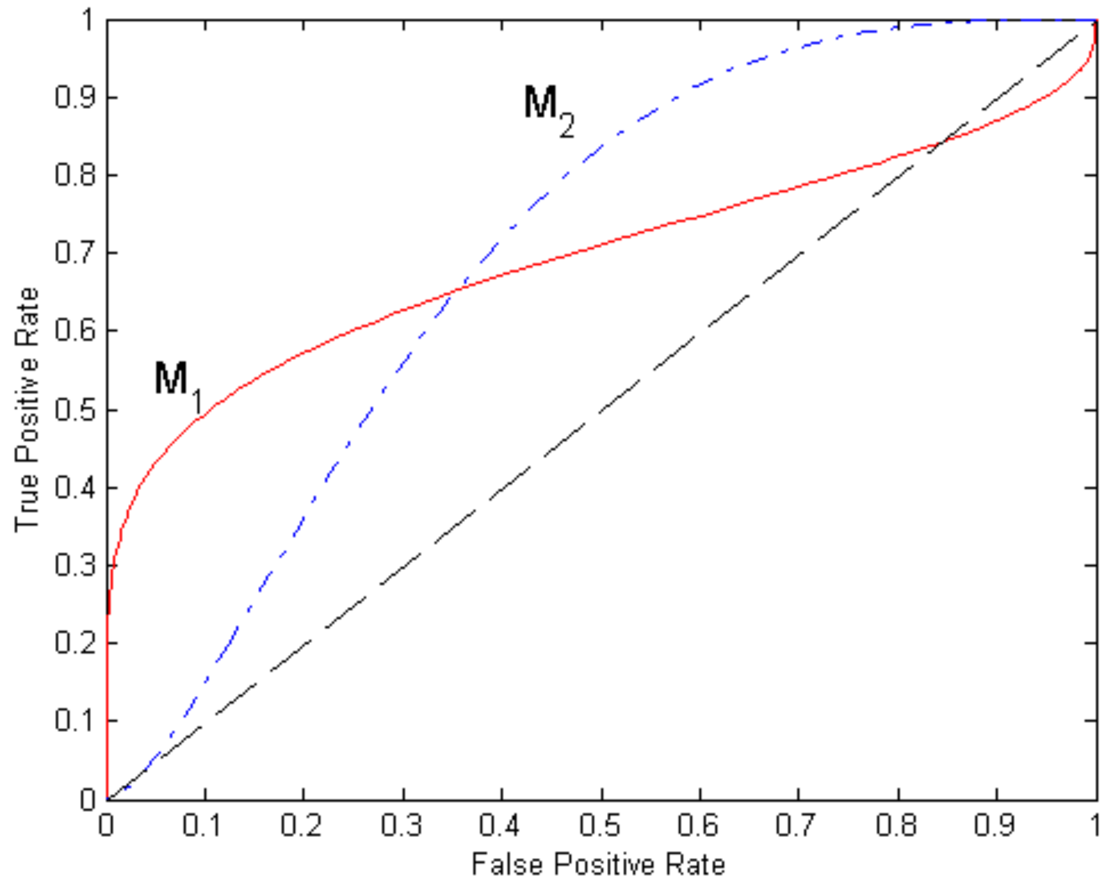
(TPR,FPR):

- (0,0): declare everything to be negative class
- (1,1): declare everything to be positive class
- (1,0): ideal

- Diagonal line:
  - Random guessing
  - Below diagonal line:
    - prediction is opposite of the true class

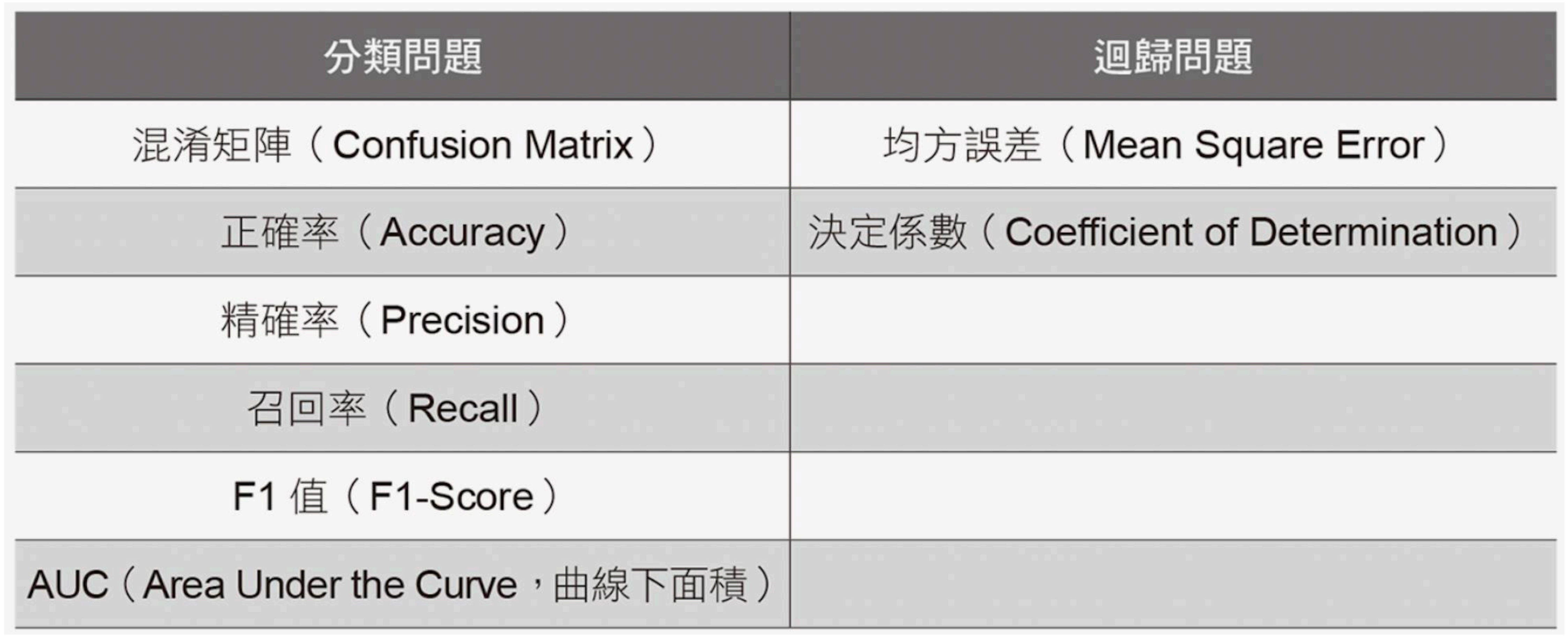


Using ROC for Model Comparison



- No model consistently outperforms the other
  - M_1 is better for small FPR
  - M_2 is better for large FPR
- Area Under the ROC curve (AUC)
  - Ideal:
    - Area = 1
  - Random guess:
    - Area = 0.5

Common Performance Metrics for Classification and Regression



分類問題	迴歸問題
混淆矩陣（Confusion Matrix）	均方誤差（Mean Square Error）
正確率（Accuracy）	決定係數（Coefficient of Determination）
精確率（Precision）	
召回率（Recall）	
F1 值（F1-Score）	
AUC（Area Under the Curve，曲線下面積）	

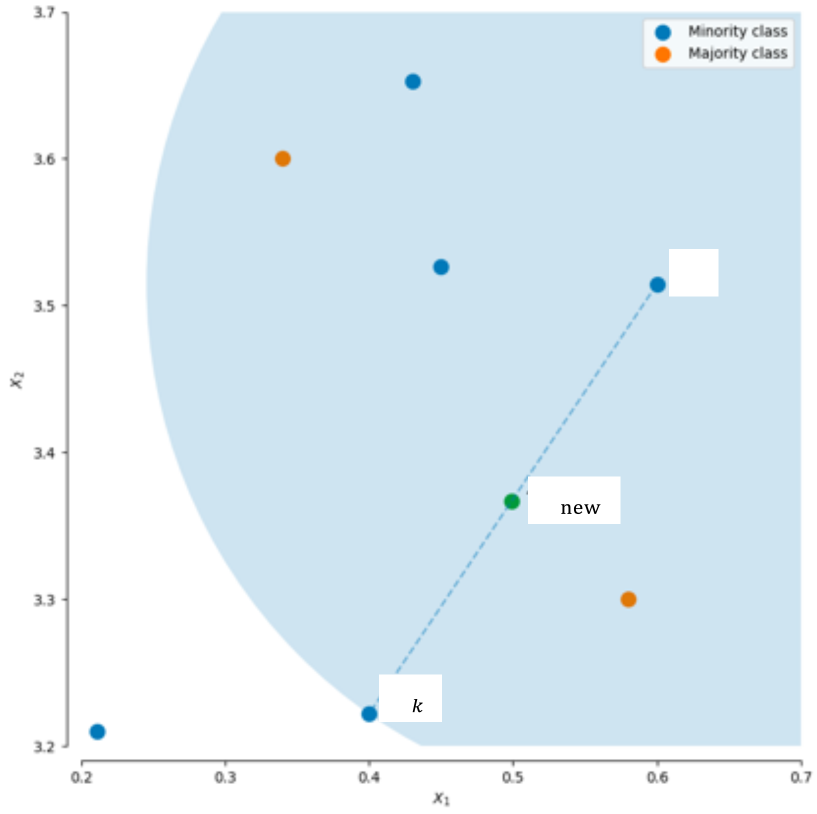
Performance_metrics.ipynb

Class Imbalance

- Choose better performance metrics
  - Accuracy is often not a good metric for class imbalance problems.
  - Use precision, recall, ROC (AUC), F-score etc.
- Balance skewed classes
  - Data oversampling and undersampling
- Cost-sensitive learning

Class Imbalance: Balance Skewed Classes

- Data Oversampling
  - Random Oversampling
  - SMOTE
  - ...
- Data Undersampling
  - Random Undersampling
  - Tomek Links
  - Edited Nearest Neighbors
  - ...
- Combined Oversampling and Undersampling



IM_data_sampling.ipynb

Class Imbalance: Cost Sensitive Algorithms

- Modified version of machine learning algorithms designed to take the differing costs of misclassification into account when fitting the model on training records.
- Many cost-sensitive algorithms to choose
  - Logistic regression
  - Support vector machines
  - Decision trees
  - Random forest
  - Gradient boosting
  - …

jupyterlab

IM_cost_sensitive_algo.ipynb

Summary: Class Imbalance Problems and Techniques

- Many practical classification problems are **class imbalance problems** where the classes are skewed (more records from one class than another)
- Sometimes you need to choose better performance metrics to train your model.
  - Accuracy is often not a good metric for class imbalance problems.
  - Use precision, recall, ROC (AUC), F-score etc.
- You can use techniques from data oversampling and undersampling (or combination) to balance the skewed class.
- **Cost-sensitive learning algorithms** are also very useful in attacking class imbalance problems.

Regression and Regression Algorithm Walkthrough

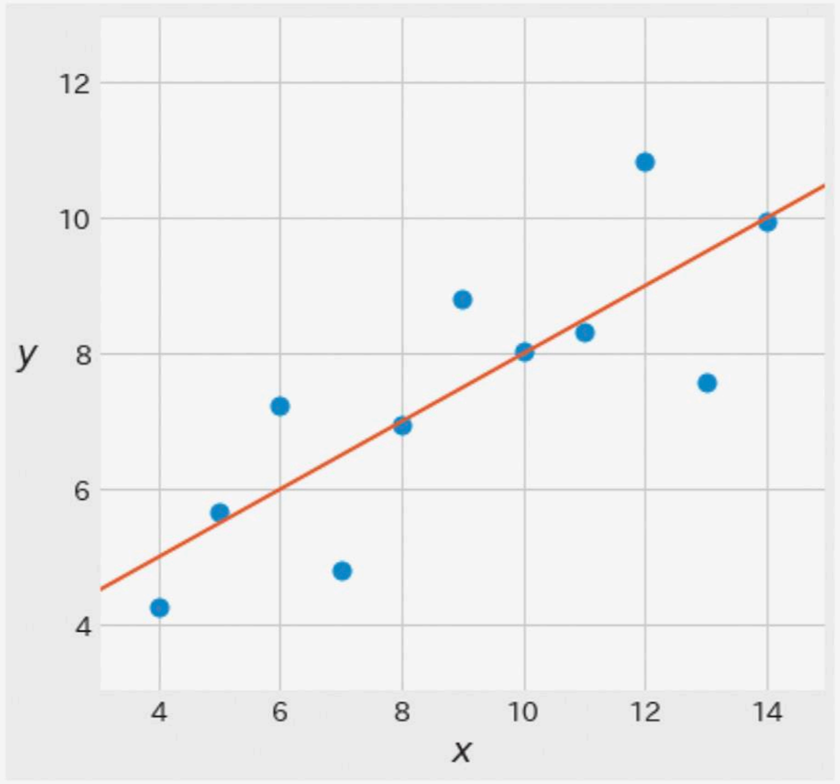
Supervised Learning: Regression

- Predict a value of a given continuous valued variable based on the values of other variables, assuming a linear or nonlinear model of dependency.
- Extensively studied in statistics, neural network fields.
- Examples:
  - Predicting sales amounts of new product based on advertising expenditure.
  - Predicting wind velocities as a function of temperature, humidity, air pressure, etc.
  - Time series prediction of stock market indices.

Simple Linear Regression

$$y \approx w_0 + w_1 x$$

- Predicting a quantitative response y on the basis of a single feature x (feature dimension = 1).

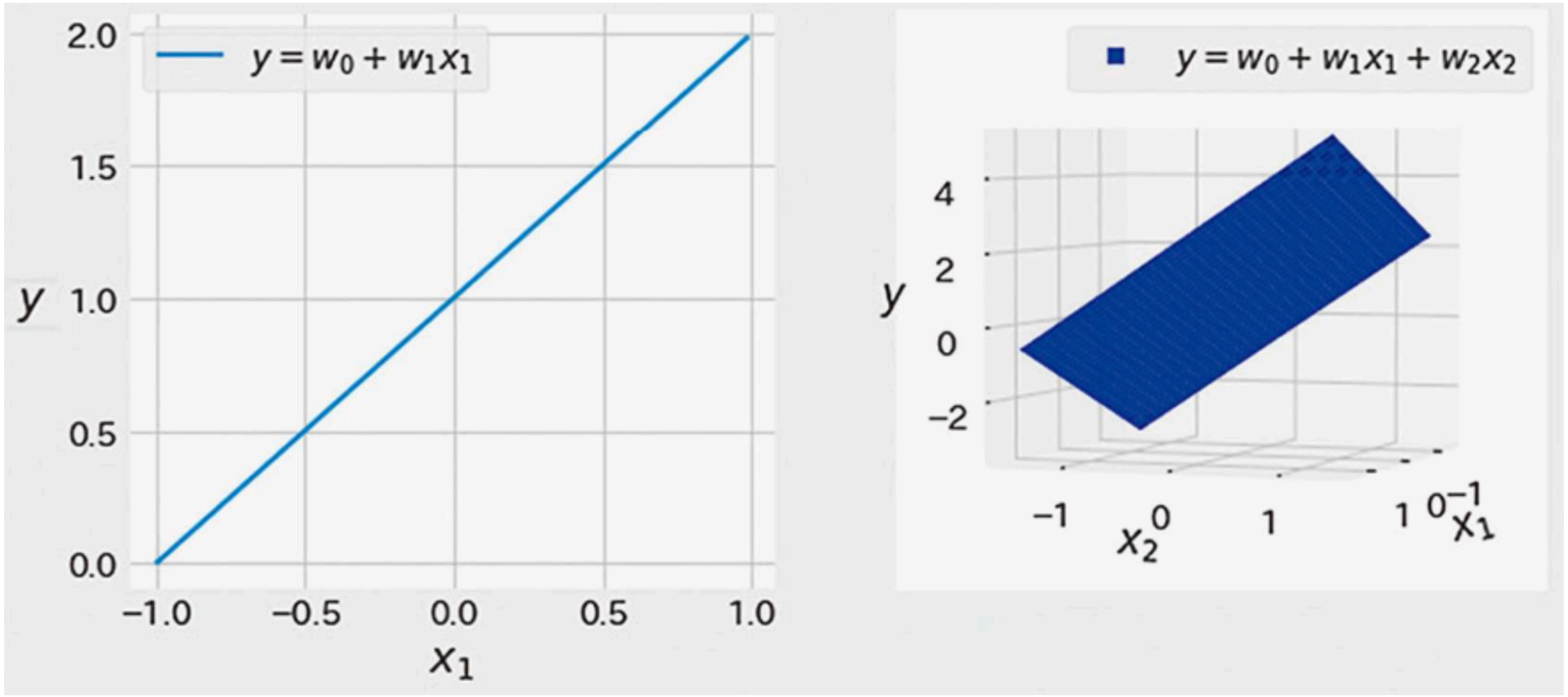


i	x	y
0	10.0	8.04
1	8.0	6.95
2	13.0	7.58
3	9.0	8.81
4	11.0	8.33
5	14.0	9.96
6	6.0	7.24
7	4.0	4.26
8	12.0	10.84
9	7.0	4.82
10	5.0	5.68

Multiple Linear Regression

$$y \approx w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_d x_d$$

- Predicting a quantitative response y on the basis of d distinct features.



Regression: Performance Metrics

- The square of sum errors (SSE) $\sum_{i=1}^{n} (y_i - \hat{y}_i)^2$
- The mean square of sum errors (MSE) $MSE = 1/n \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$
- R^2, Coefficient of Determinant $R^2 = 1 - \frac{SSE}{TSS}$

$$TSS = \sum_{i=1}^{n} (y_i - \bar{y}_i)^2$$

measures **the total variance** in the response y, and can be thought of as the amount of variability inherent in the response before the regression is performed.

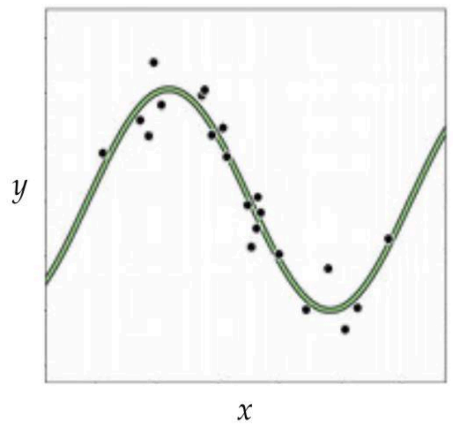
An R^2 statistic measures the proportion of variability in y that can be explained using x. An R^2 statistic that is close to 1 indicates that a large proportion of the variability in the response has been explained by the regression.

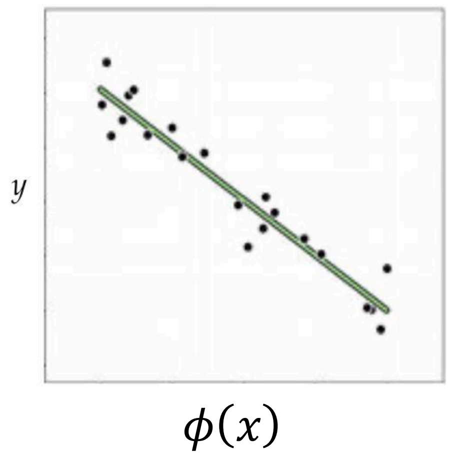
Linear_Regression.ipynb

Nonlinear Regression: Feature Transformation, Basis Functions, Feature Exploration and Kernel Method

Nonlinear Regression: Feature Transformation and Basis Function

- single feature x (feature dimension = 1).





$$y \approx w_0 + w_1 \phi(x)$$

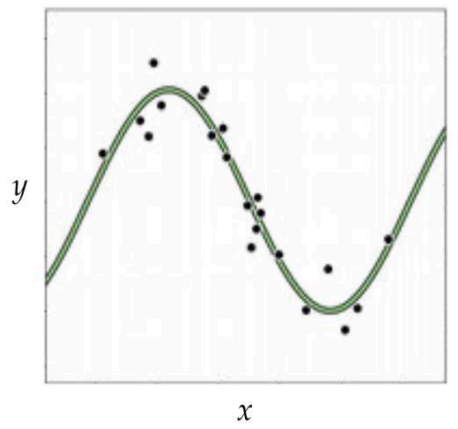
$$\phi(x) = \sin(\alpha_0 + \alpha_1 x)$$

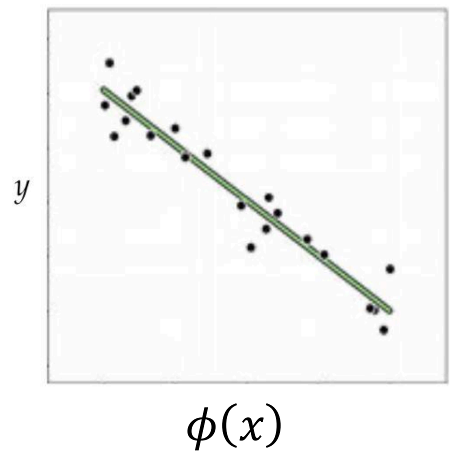
- **Important observation**: $y \approx w_0 + w_1 \phi(x)$ remains linear with respect to w_1
- $\phi(x)$ is called **feature transformation**
- To apply feature transformation systematically, we often use some **basis functions** for $\phi(x)$.

See Feature_Transformation.pdf

Nonlinear Regression: Single vs. Multiple Features

- single feature x (feature dimension = 1).





$$y \approx w_0 + w_1 \phi(x)$$

$$\phi(x) = \sin(\alpha_0 + \alpha_1 x)$$

- multiple features

$$y \approx \overline{w}_0 + \overline{w}_1 \phi(x_1) + \overline{w}_2 \phi(x_2) \qquad \phi(x) = (\alpha_0 + \alpha_1 x + \alpha_2 x^2)$$

$$y \approx w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_1 x_2 + w_5 x_2^2$$

we are now dealing with five-dimensional instead of two-dimensional feature space.

Nonlinear Regression: Feature Explosion

Fun Time: let the input dimension d is of moderate size, e.g., $d = 100$, then the associated polynomial degree $n = 5$, what would be the resulting feature dimension? (1) ~ 10^3 (2) ~ 10^5 (3) ~ 10^7 (4) ~ 10^9

Nonlinear Regression: Feature Explosion

Fun Time: let the input dimension d is of moderate size, e.g., $d = 100$, then the associated polynomial degree $n = 5$, what would be the resulting feature dimension? (1) ~ 10^3 (2) ~ 10^5 (3) ~ 10^7 (4) ~ 10^9

- The precise dimension of the resulting feature m after applying n^{th}order polynomial transformation for d distinct input features is:

$$m = \frac{(d+n)!}{d!\,n!} - 1 \qquad\qquad m \approx O(d^n)$$

- For the input dimension $d = 100$ and $d = 1000$, the associated polynomial degree $n = 5$ would have resulting features of $m =$ 96,560,645 and 8,459,043,543,950, respectively.

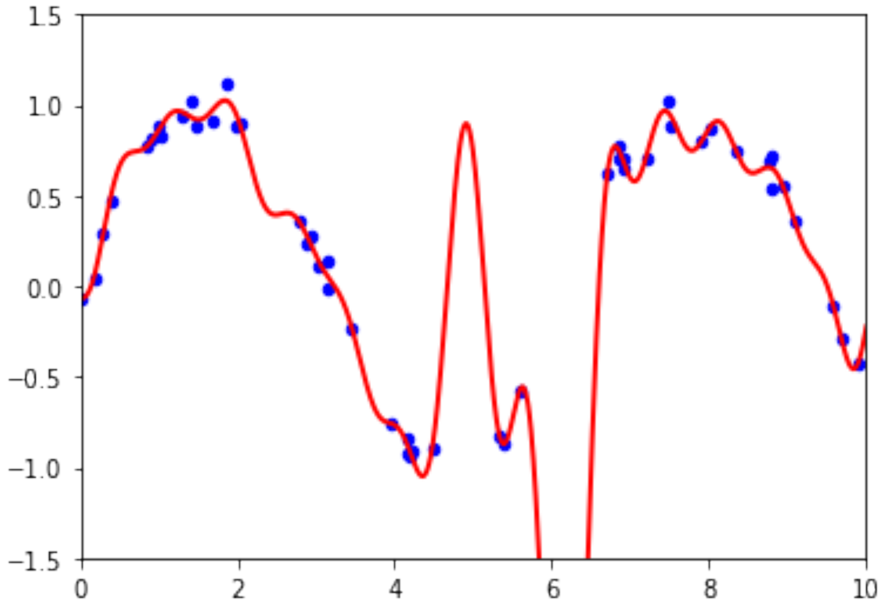
Nonlinear Regression: Feature Explosion

- **The exponential growth challenge of the feature dimension after feature transformation $\mathbf{x} \rightarrow \phi(\mathbf{x})$ can be cleverly solved by the kernel-based method.**
- **A kernel is just a form of generalized dot product $\phi(\mathbf{x}) \cdot \phi(\mathbf{z})$**
- **The key insight in kernel-based method is that you can rewrite many linear models in a way that doesn't require you to ever explicitly compute $\phi(\mathbf{x})$**
- **The topic is often called kernel tricks, a very important topic in ML but beyond the scope of this course.** (refs: Kernel_Trick01.pdf, Kernel_Trick02.pdf)

Nonlinear Regression: Overfit and Regularization

Regularization

- High-order nonlinear regression can easily lead to **overfit**.



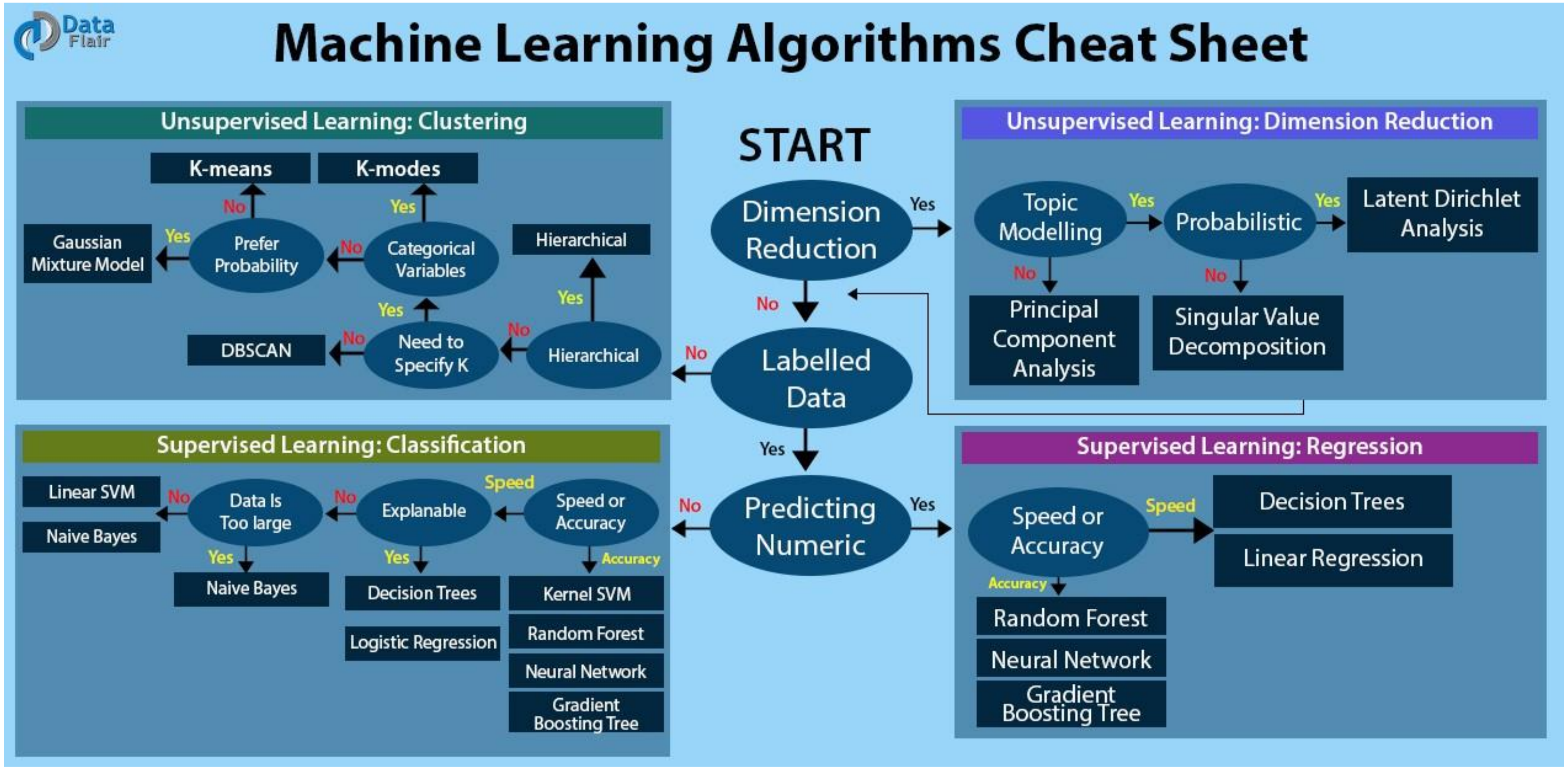
- **Regularization** can help!
  - Ridge (L2) regularization
  - Lasso (L1) regularization

Regularization.pdf

Summary: Regression

- Regression is used to predict a continuous value based some given features.
- Feature transformation $x \to \phi(x)$ allows us to extend linear models to nonlinear models.
- The exponential growth challenge of the feature dimension after feature transformation $x \to \phi(x)$ can be cleverly solved by the **kernel**-based method.
- The kernel method is also used in SVM.
- Nonlinear regression tends to overfit and ridge (L2) and lasso (L1) regularization techniques are often used to reduce overfit.
- Many classification methods such as SVM, decision tree, ensemble trees can also handle regression problems.

Cheat Sheet (備忘表)



https://medium.com/dataflair/beat-the-heat-with-machine-learning-cheat-sheet-365c25bd1c3