supervised learning: learning model from label
unsupervised learning: extract meaningful information without label (clustering / dimensionality reduction)
Reinforced learning: improve the performance based on interaction with environment. The feedback is a measure of how well the action was measured by a reward function.

Decision Tree	non
Linear SVM	linear
SVM with kernel	non
Linear Regression	a classifier
Logistic Regression	linear
Naive Bayes	linear

Linear Regression: Least Square Fitting

minimize: $\sum_i^n [y_i - (\beta_0 + \beta_1 x_i)]^2$

$$S_{xy} = \sum_{i=1}^n x_i y_i - \frac{1}{n}\left(\sum_{i=1}^n x_i\right)\left(\sum_{i=1}^n y_i\right)$$

$$S_{xx} = \sum_{i=1}^n x_i^2 - \frac{1}{n}\left(\sum_{i=1}^n x_i\right)^2$$

$$S_{yy} = \sum_{i=1}^n y_i^2 - \frac{1}{n}\left(\sum_{i=1}^n y_i\right)^2$$

$$\hat\beta_0 = \bar{y} - \hat\beta_1 \bar{x} \qquad \hat\beta_1 = \frac{S_{xy}}{S_{xx}}$$

Multiple: $\hat\beta = (X^T X)^{-1} X^T Y$

Decision Trees:

when have N attributes, $2^{(2^N)}$ trees

Choose feature $X_i = H(Y|X_i)$ ↓

$I(X;Y) = H(Y) - H(Y|X)$ ↑

$H(X) = -\sum p(x) \log_2 p(x)$

$H(Y|X) = \sum p(x) H(Y|X=x)$

$I(X,Y) = \sum\sum P_{X,Y}(x,y) \log\left(\frac{P_{X,Y}(x,y)}{P_X(x) P_Y(y)}\right)$

Gini $= 1 - \sum p_j^2$

classification Error $= 1 - \max p_j$

Decision Tree Depth ↑ → overfitting

Ensemble learning - Random Forest

reduce overfitting and variance without decreasing performance

Bagging - Bootstrap Aggregating

Bootstraping: Random sampling with replacement

train multiple decision trees & search all features to split on for each tree

Aggregating: Combine multiple predictions via averaging or majority vote

Support Vector Machines

kernel: mapping to higher-dimensional space

complexity depends on the number of training samples, not dimensionality

larger margin — lower error

$\phi(w) = w^T w$ is minimized

Soft margin is allowed errors.

$\phi(w) = w^T w + \boxed{C} \sum \xi_i$

penalty $C \to 0$ underfitting

$C \to \infty$ overfitting

SVM Weakness:

1. sensitive to noise
2. Standard SVM only consider 2 classes ↳ build multiple SVMs
3. select a specific kernel and parameters is usually done by see and try

Naive Bayes → MAP

$$\arg\max_y P(y|x) = \arg\max_y P(y) \prod_{i=1}^n P(x_i|y)$$

{ Logistic Regression directly compute $P(y|x)$
Naive Bayes use Bayes Theorem to compute $P(y|x)$

NB + Gaussian Basis Function

≈ LR + sigmoid

Logistic Regression model the $P(y|x)$ as a logistic function. The logic is a weighted linear combination of the features. linear regression itself is based on linear feature function to regress.

→ Or using total probability:
P(S)=P(S|yes)P(yes)+P(S|no)P(no) = (2/9)(9/14) + (3/5)(5/14) = 5/14

→ Naïve Bayes: P(yes|S,W) = P(S,W|yes)P(yes)/P(S,W) = (6/81)(9/14)/(37/210) = 10/37
→ Naïve Bayes: P(S,W|yes) = P(S|yes)P(W|yes) = (2/9)(3/9) = 6/81
→ P(S,W)=P(S,W|yes)P(yes) + P(S,W|no)P(no)
= P(S|yes)P(W|yes)P(yes) + P(S|no)P(W|no)P(no)
= (2/9)(3/9)(9/14) + (3/5)(3/5)(5/14) = 6/126 + 9/70 = 37/210

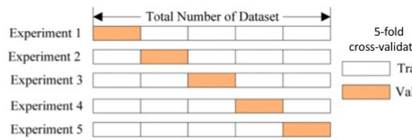
Performance Metrix

	Actual value	
Predicted value	TP	FP
	FN	TN

Recall: $\frac{TP}{FN+TP}$ when false negatives is catastrophic. e.g. disease detection

Precision: $\frac{TP}{TP+FP}$ when being right (positive prediction is correct) outweighs detecting all positives. e.g. recommendation system

1. Training set: ↓
2. Validation set: hyperparameters tuning and model selection ↓
3. Test data

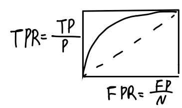


K-fold cross validation is used when we have little data

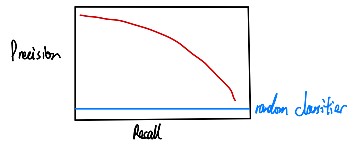
F1-score: $2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$

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

ROC Curves: the trade-off between TP rate and FP rate



PR Curve: between TP rate and positive predictive value



Occam's Razor

Given 2 models with similar generalization errors, the simple model is preferred.