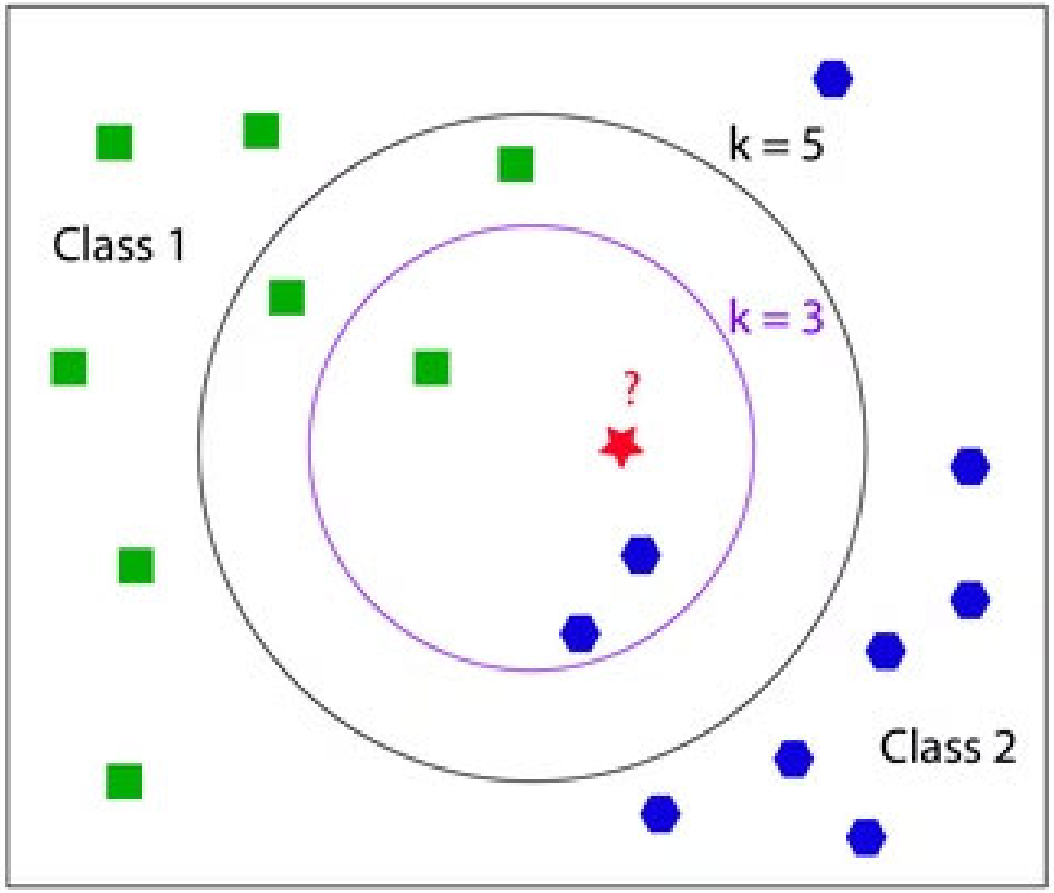
K-Nearest Neighbour

Classification through proximity-based decision making



k-Nearest Neighbour Algorithm

Core Principle

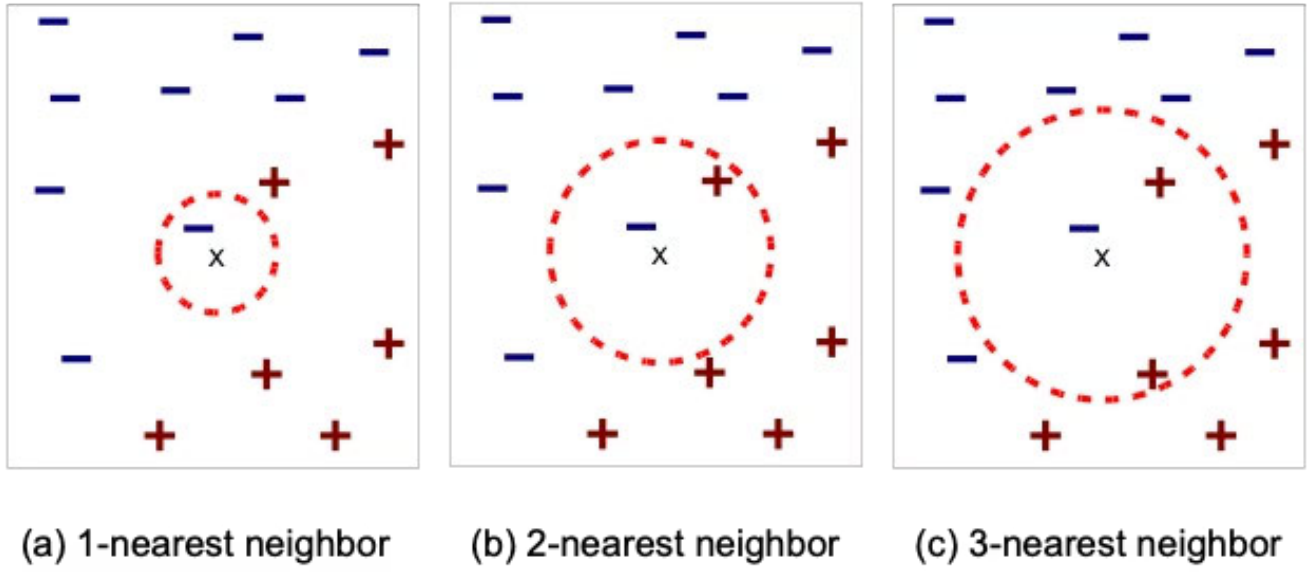
Assign observation to class of nearest neighbour(s)

Decision Method

Use majority voting when k > 1

Distance Metric

Euclidean distance computationally intensive for all pairs



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

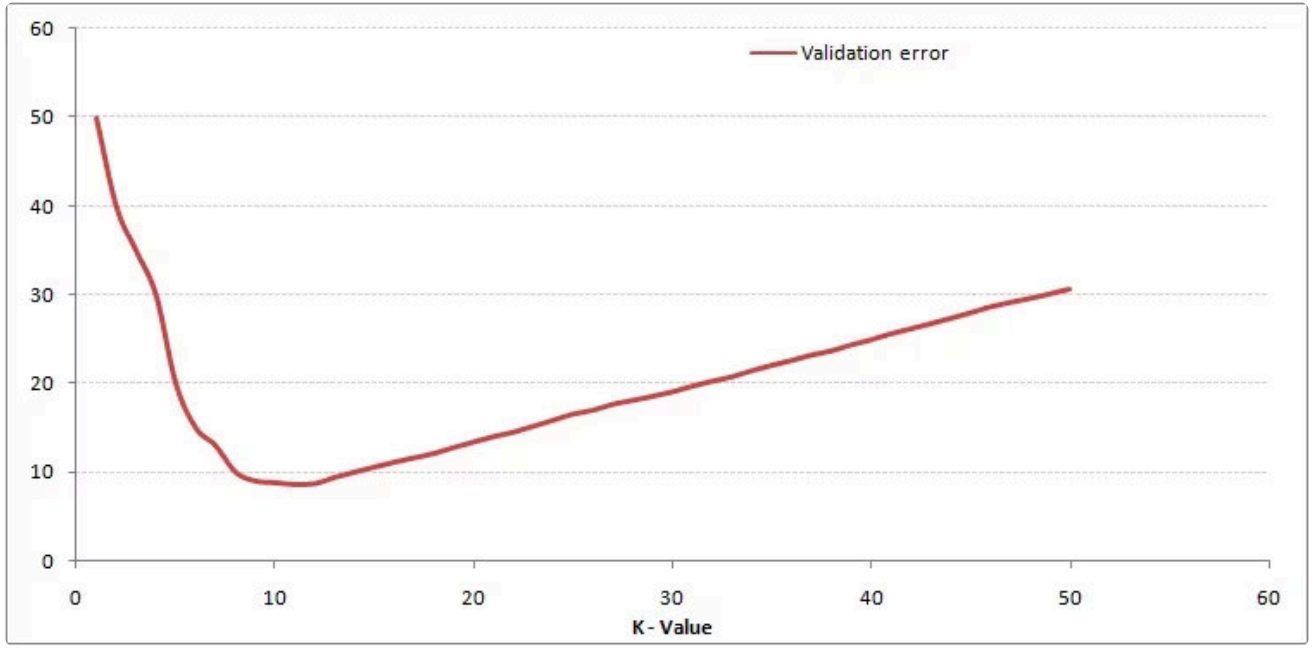
Selecting the Optimal k Value

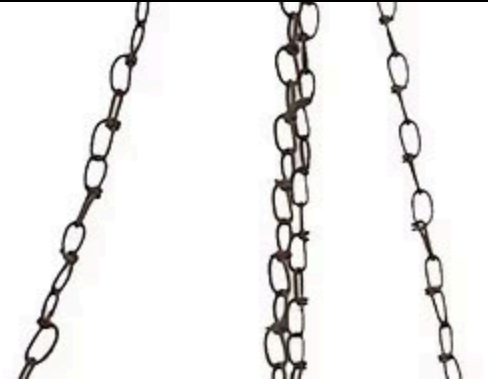
Create validation curves to identify the k value that minimizes error

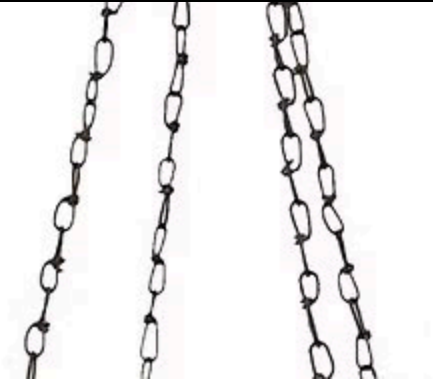
Balance between underfitting and overfitting

- Lower k: more sensitive to noise
- Higher k: smoother decision boundaries
- Optimal k varies by dataset characteristics

=> Make this plot for your data.







Characteristics of k-NN

1

Straightforward algorithm

Simply assign to nearest neighbour or majority voting

2

Parameter Sensitivity (k)

Performance heavily dependent on k selection:

- Too small, overly sensitive to noise/outliers
- Too large, neighbours include points from other classes

3

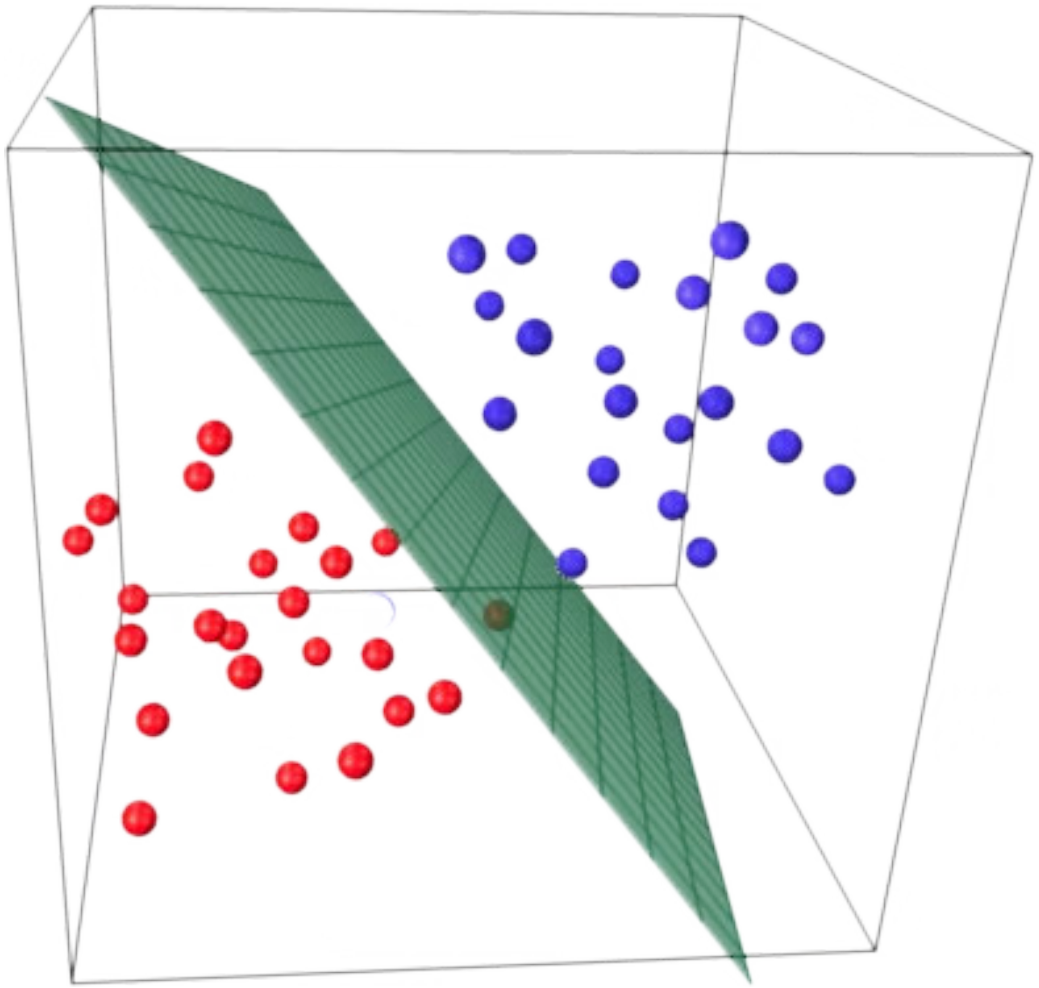
Computational Cost

Requires distance calculation to all training points. Can be sped up, e.g. parallel computing

4

Memory Requirements

Must store entire training dataset in memory



Logistic Regression

Binary Classification Framework

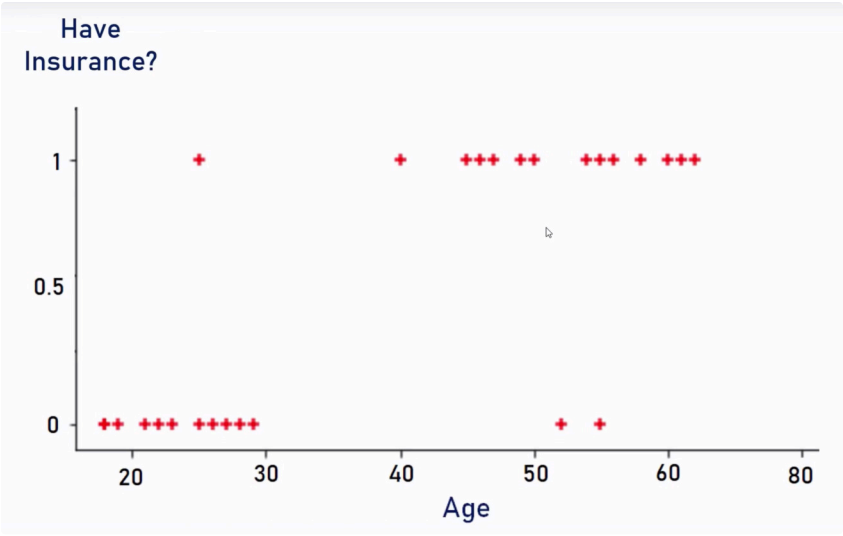
age	have_insurance
22	0
25	0
47	1
52	0
46	1
56	1
55	0
60	1
62	1
61	1
18	0
28	0
27	0
29	0
49	1

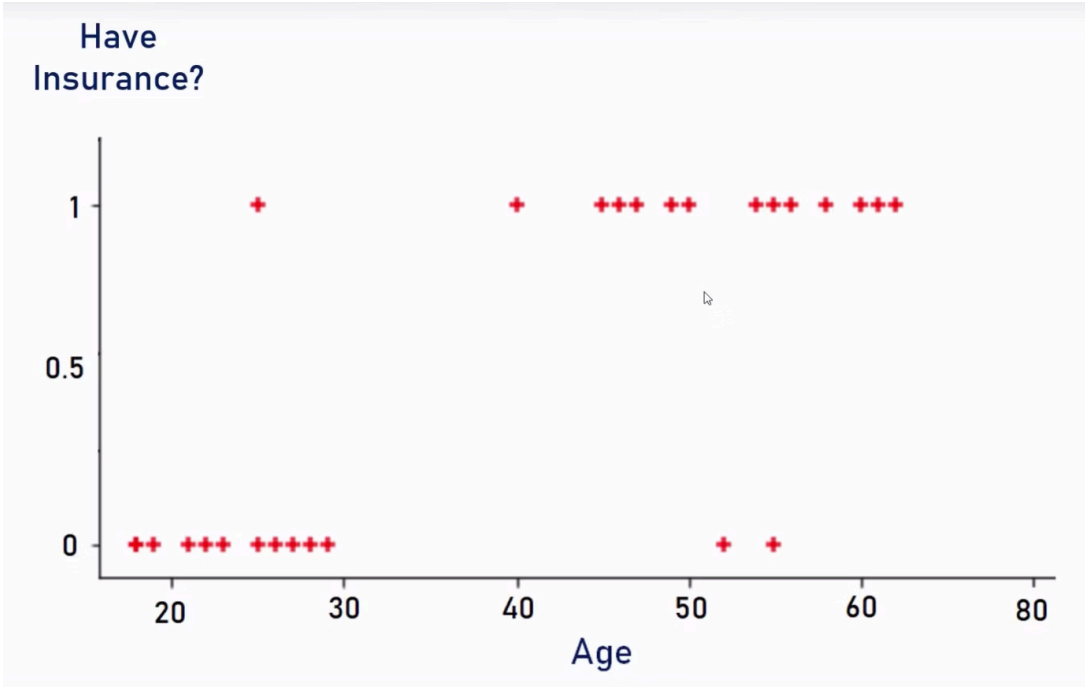
Two-Class Problem

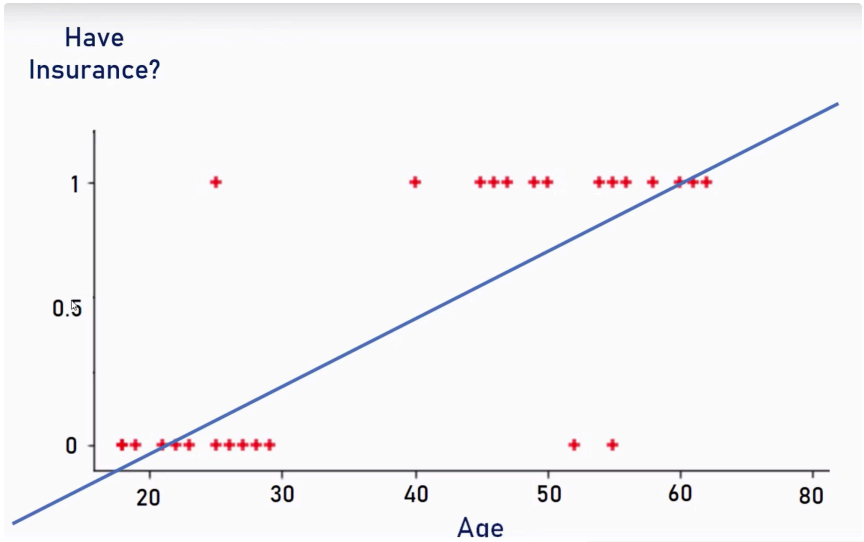
Distinguishing between two mutually exclusive categories

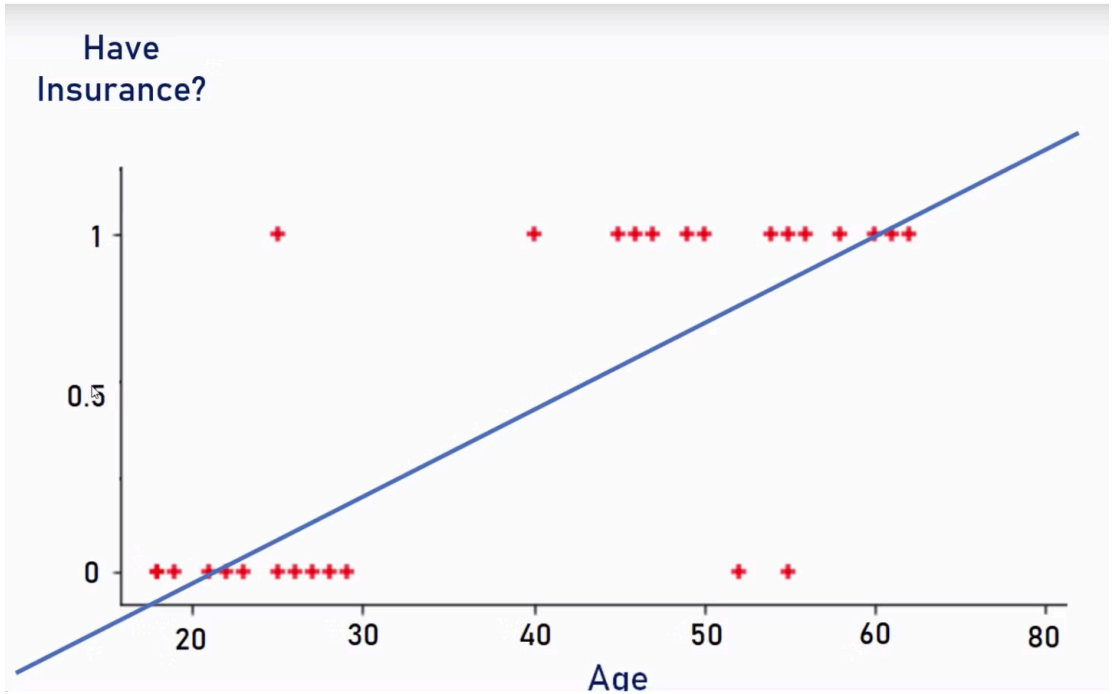
- Output: probability of class membership
- Decision boundary separates classes
- Threshold determines final classification

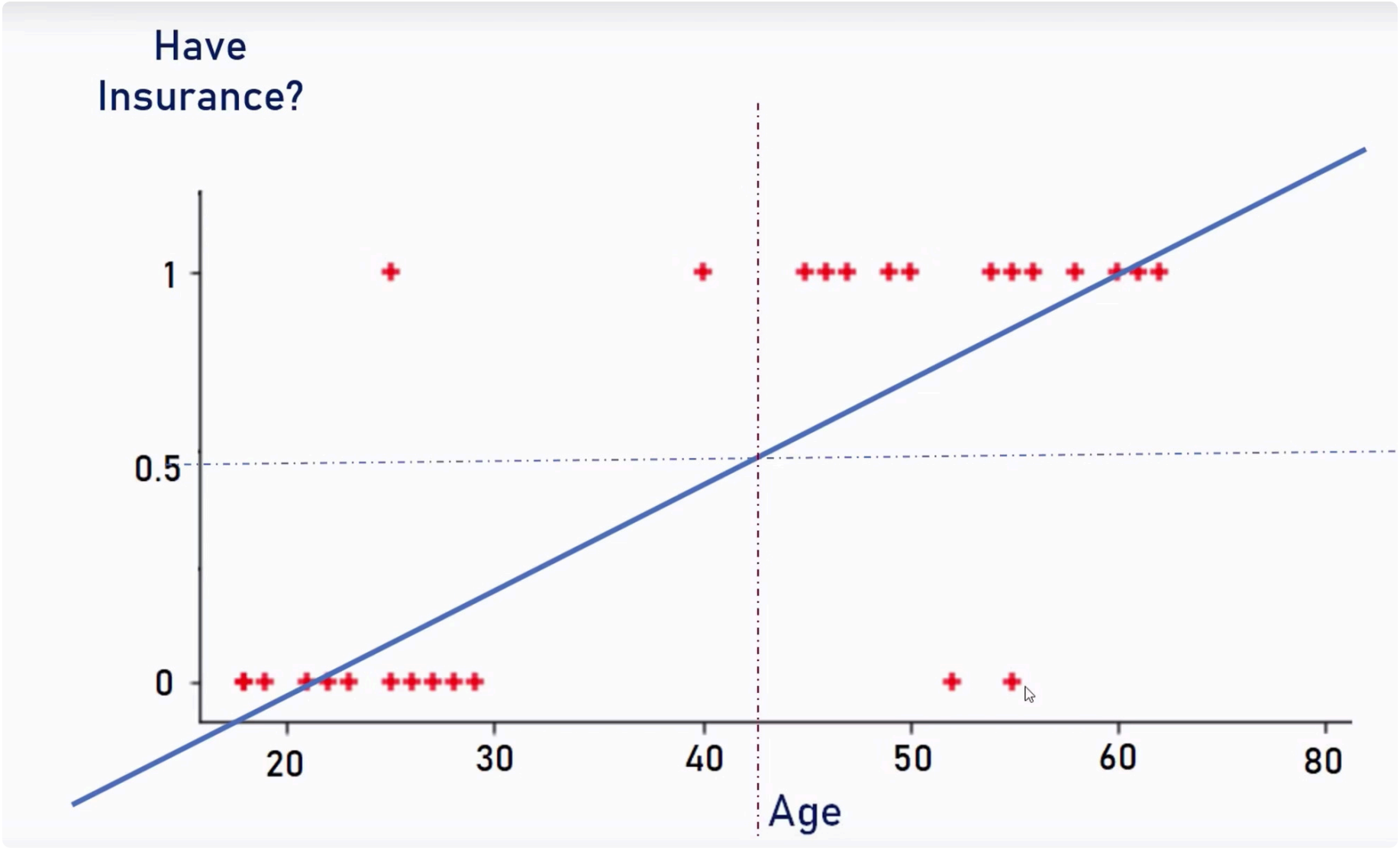
Visual explanation

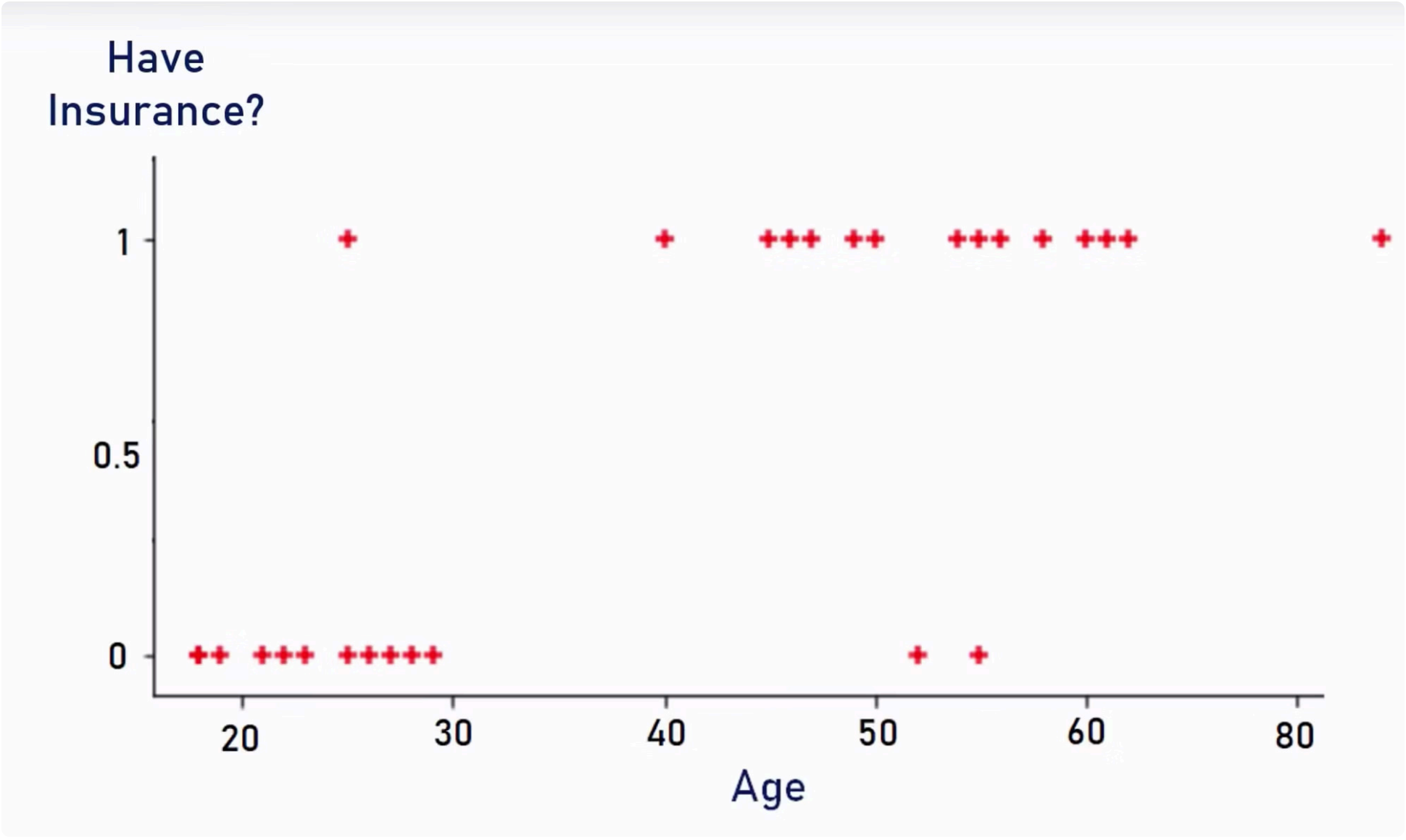

Have
Insurance?
1
0.5
0
20
30
40
50
60
80
Age

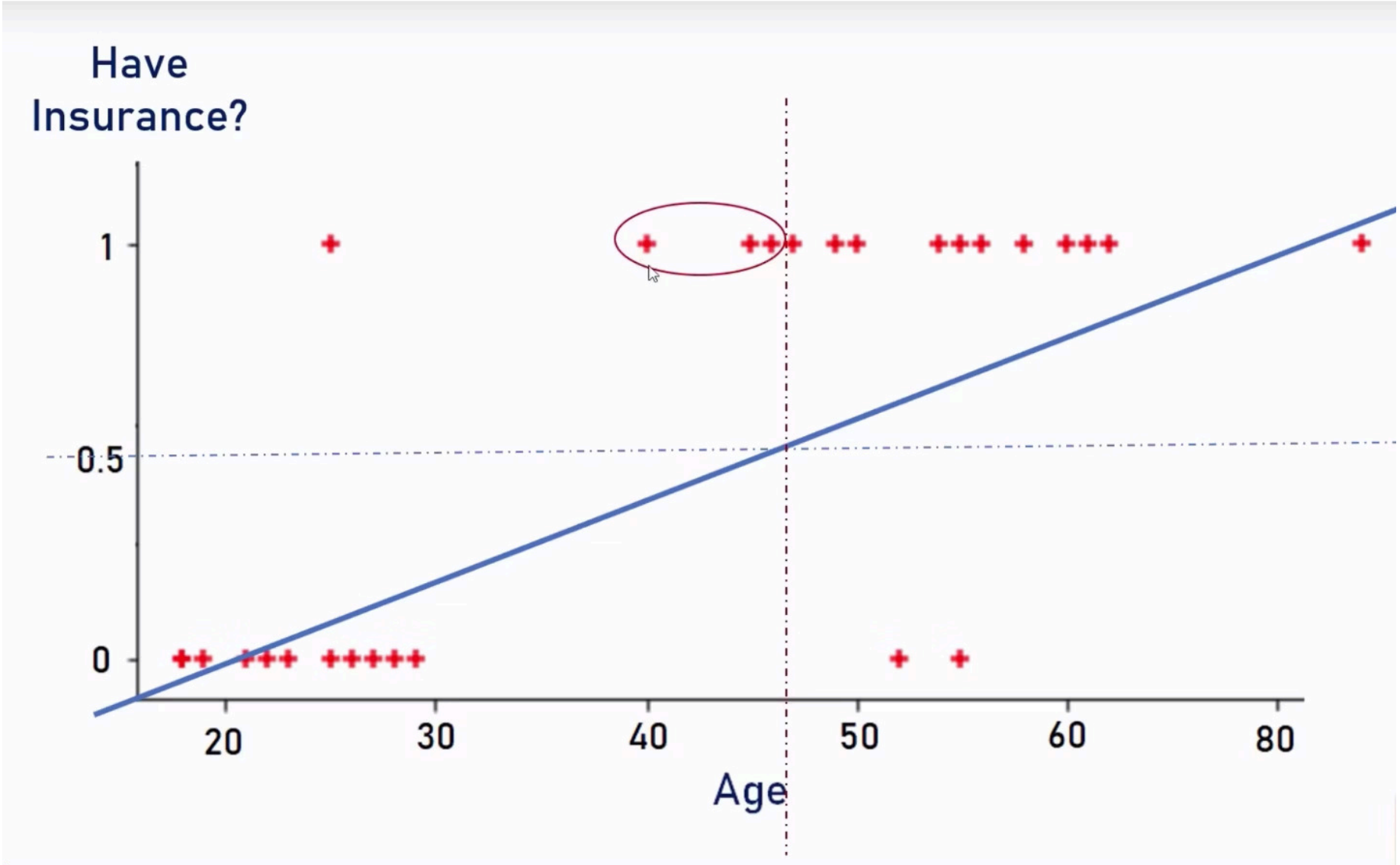

Have
Insurance?
1
0.5
0
20
30
40
50
60
80
Age

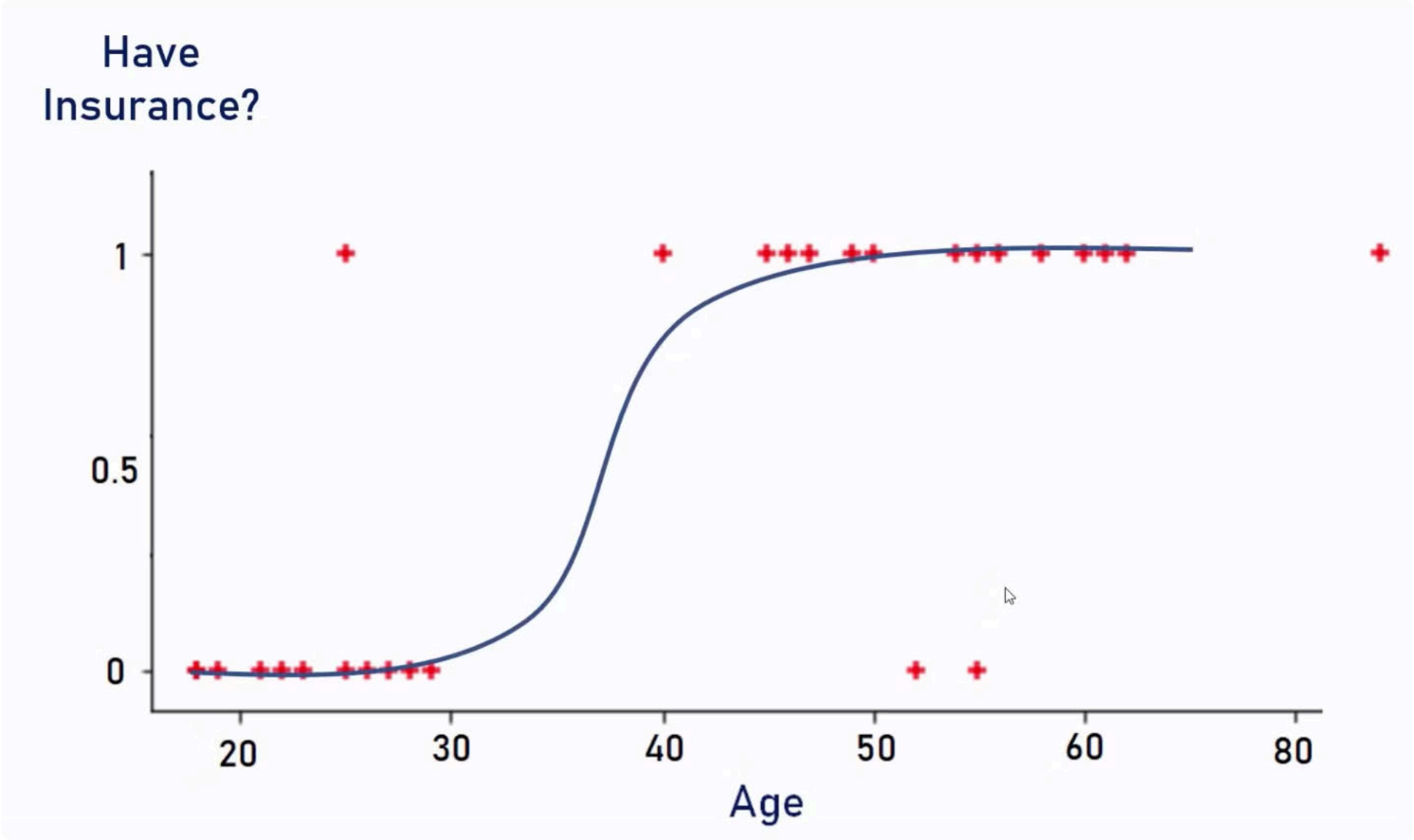

Have
Insurance?
1
0.5
0
20
30
40
50
60
80
Age

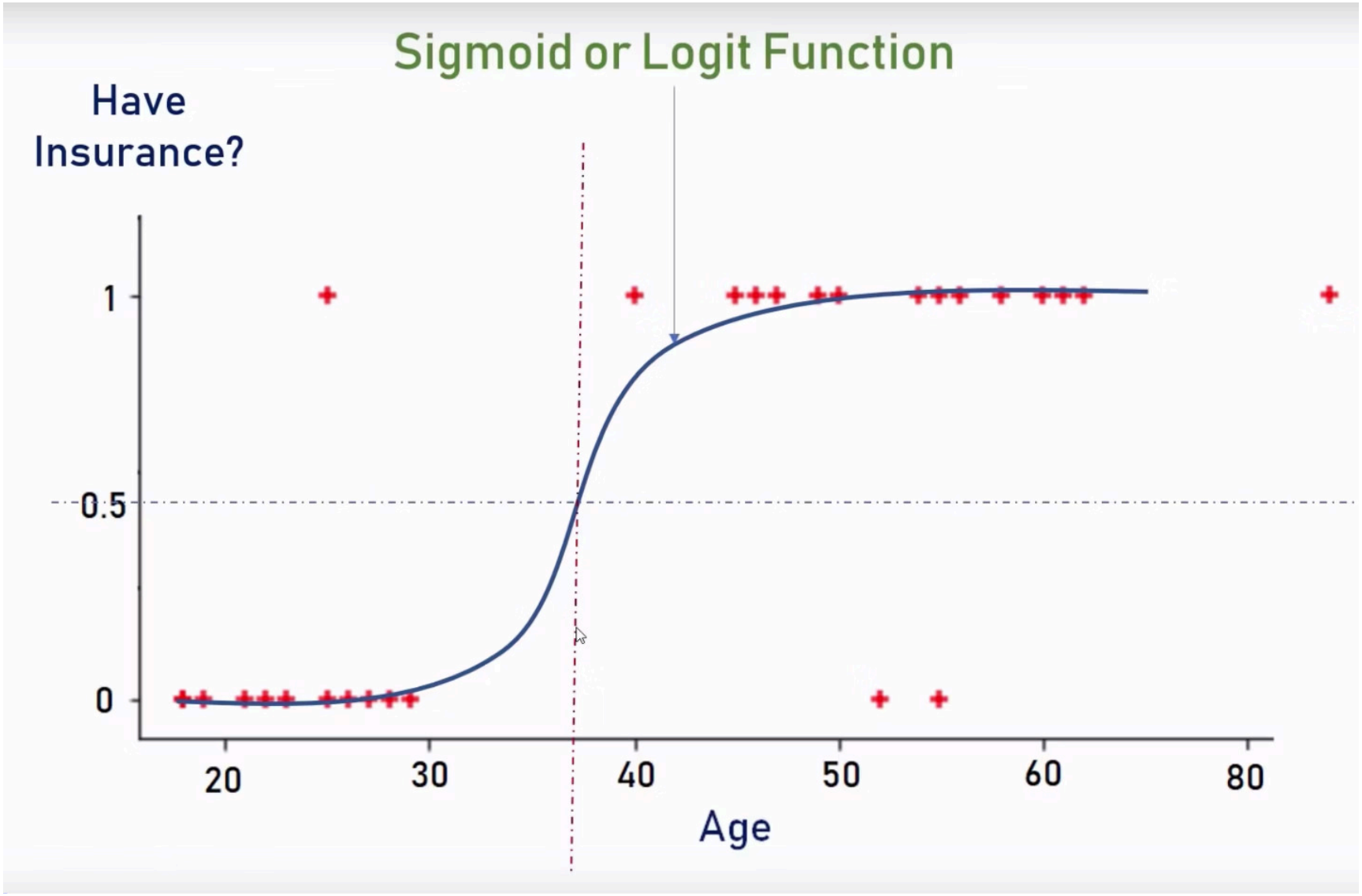

Have
Insurance?
1
0.5
0
20
30
40
50
60
80
Age


Have
Insurance?
1
0.5
0
20
30
40
50
60
80
Age


Have Insurance?
1
0.5
0
20
30
40
50
60
80
Age

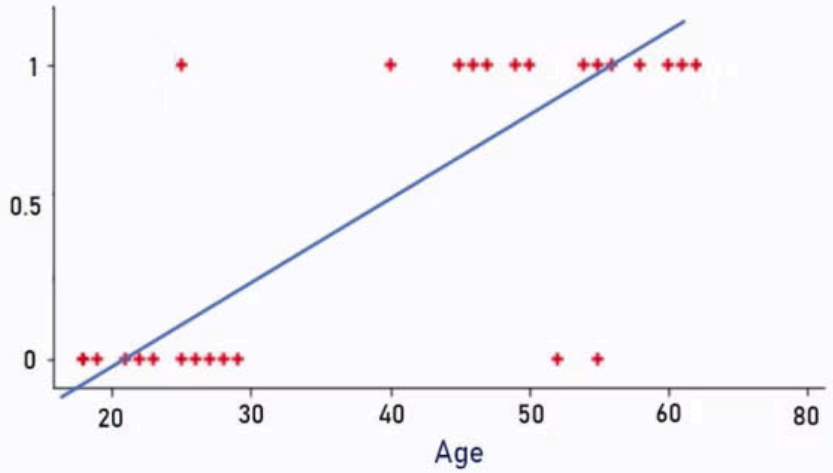

Have
Insurance?
1
0.5
0
20
30
40
50
60
80
Age

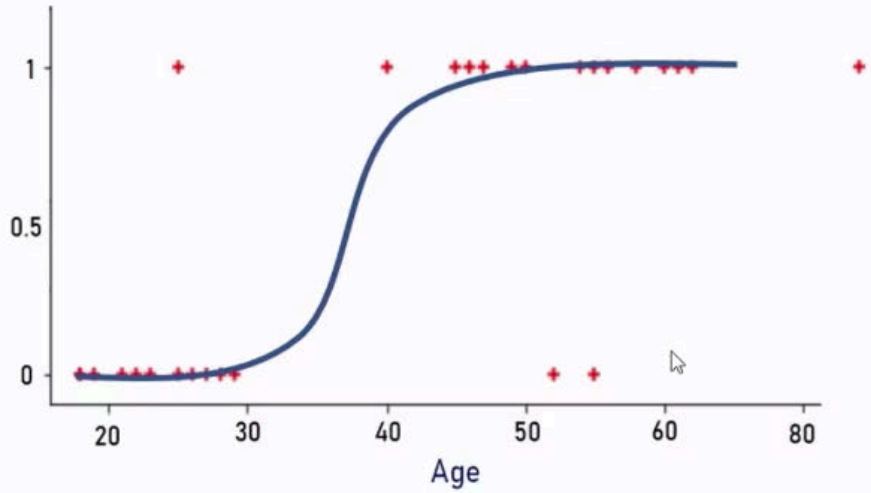

Have
Insurance?
1
0.5
0
20
30
40
50
60
80
Age


Sigmoid or Logit Function
Have Insurance?
1
0.5
0
20
30
40
50
60
80
Age

$$y = m * x + b$$

$$y = \frac{1}{1 + e^{-(m*x+b)}}$$





Optimizing Threshold Values

Always 0.5?

No. Default threshold isn't always optimal

- Analyze confusion matrix patterns
- Consider cost of false positives vs false negatives
- Adjust based on domain requirements
- Balance precision and recall tradeoffs

		Actual	
		Positive	Negative
Predicted	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

Logit

The link function that transforms probabilities into a linear form

Logistic regression (multiple variables):

- $P(\text{class is } 1) = \frac{1}{(1+e^{(-(m1*x1+m2*x2+\ldots+b))})}$
- $P(\text{class is } 0) = 1 - \frac{1}{(1+e^{(-(m1*x1+m2*x2+\ldots+b))})}$

Logit:

Class is 1 when:

Odds = $\frac{P(\text{class is } 1)}{P(\text{class is } 0)} = e^{((m1*x1+m2*x2+\ldots+b))} > 1$

Logit(P): $log(Odds) = m1 * x1 + m2 * x2 + \ldots + b > ln(1) = 0$ ~ linear regression

Deriving the Logit Function

Mathematical foundation connecting linear predictors to probability space

$$\frac{P(\text{class } 1)}{P(\text{class } 0)} = \frac{\frac{1}{1+e^{-z}}}{1-\frac{1}{1+e^{-z}}} = \frac{\frac{1}{1+e^{-z}}}{\frac{(1+e^{-z})-1}{1+e^{-z}}}$$

$$= \frac{\frac{1}{1+e^{-z}}}{\frac{e^{-z}}{1+e^{-z}}} = \frac{\frac{1}{1+e^{-z}}}{e^{-z}} \cdot (1+e^{-z}) = \frac{1}{e^{-z}} = e^{z}$$

with $z = m_1 x_1 + m_2 x_2 + \dots b$

Multiclass Classification with LR

One-vs-Rest Approach

Build **separate** regression model for each class

Prediction Method

Assign to class with highest prediction score

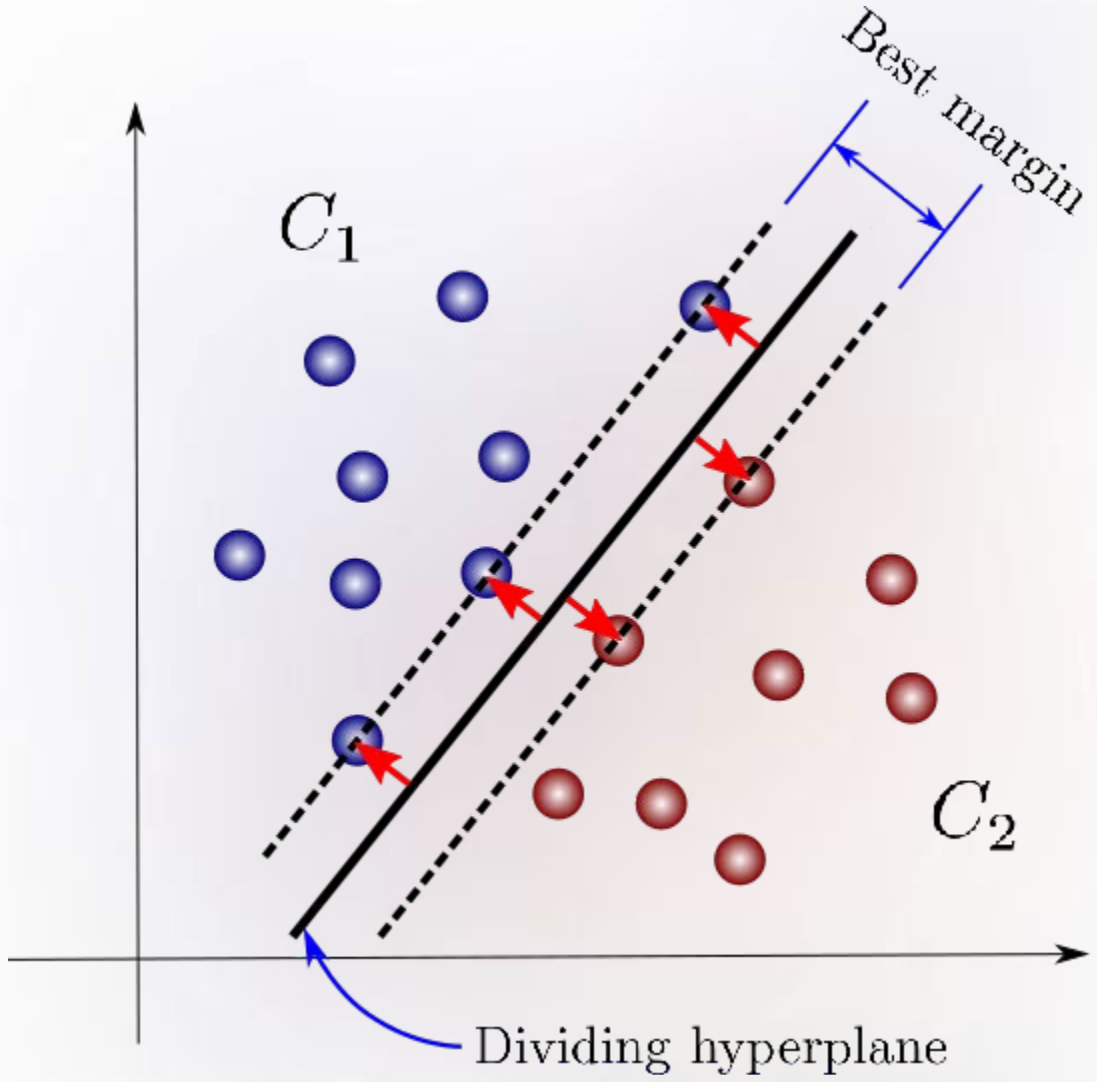
Limitation

Models built independently—no inter-class dependencies captured

Example: Categories "cheap," "expensive," "car," "tree"—detecting both "car" and "expensive" doesn't inform each model of the other's prediction

Support Vector Machines

Maximum margin classifiers for optimal decision boundaries



SVM: Historical Context

Origins

Statistical learning theory by Vapnik et al., 1992

Breakthrough Performance

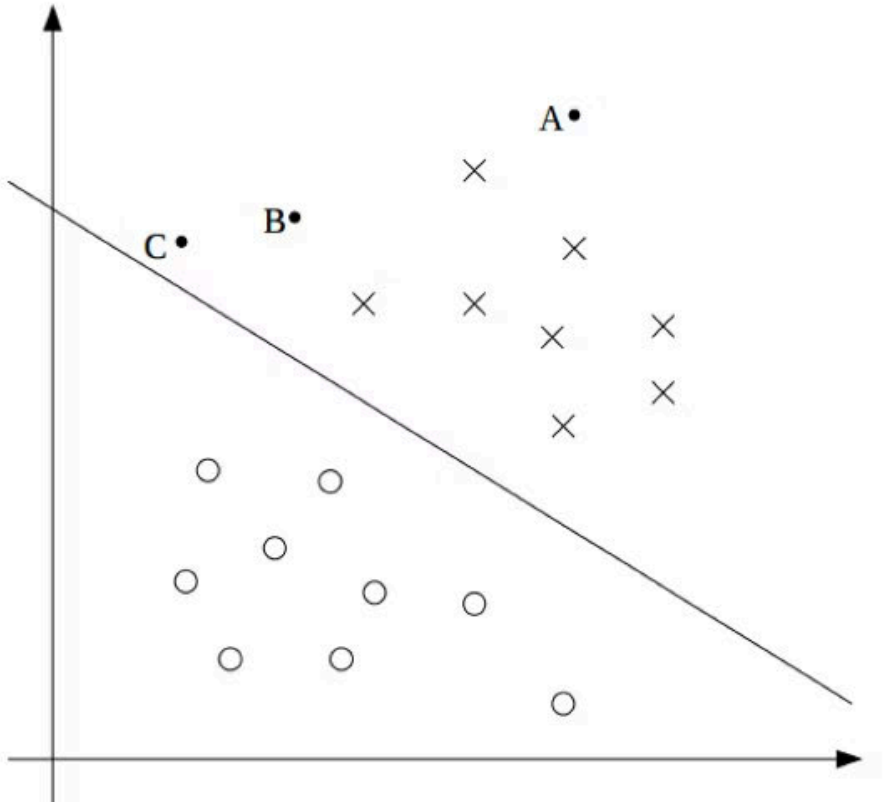
Performed as well as neural networks on handwriting recognition tasks

Applications Across Domains

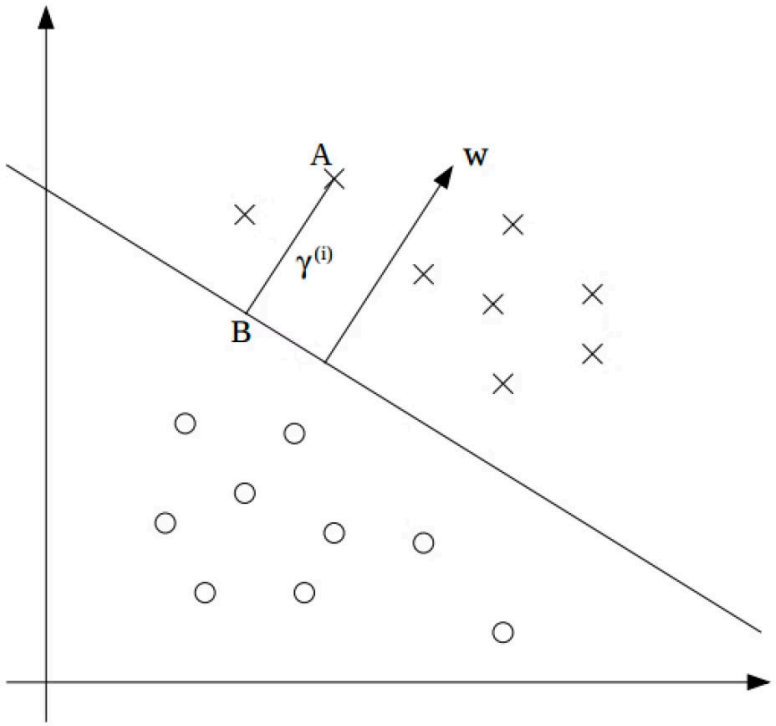
- Object detection and recognition
- Content-based image retrieval
- Text and speech recognition
- Biometric identification systems

Margins and Classification Confidence

Wider margins indicate greater confidence and better generalization to unseen data



Geometric Margins

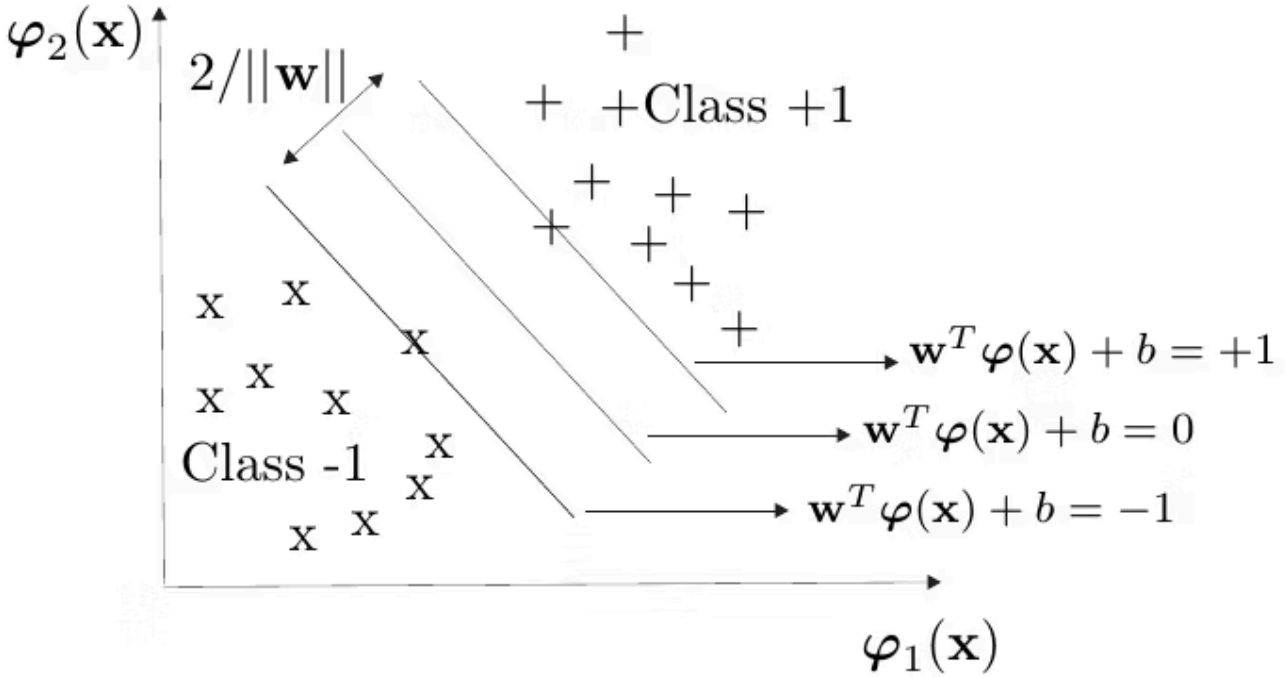


(Ng, 2014)

Distance from data points to the separating hyperplane

- Perpendicular distance measurement
- Scale-invariant metric
- Determines classification robustness
- Foundation for margin maximization

Optimal Margin Classification



Support Vectors

Data points lying closest to the decision hyperplane

These critical points define the optimal separating boundary and maximum margin

Only **support vectors** influence the final model—other points are irrelevant

Optimal margins

Given a training set of N data points $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ with input data $\mathbf{x}_i \in \mathbb{R}^n$ and corresponding binary class labels $y_i \in \{-1, +1\}$, the SVM classifier should fulfil following conditions. (vapnik, 1995):

$$\begin{cases} \mathbf{w}^T \varphi(\mathbf{x}_i) + b \geq +1, & \text{if } \; y_i = +1 \\ \mathbf{w}^T \varphi(\mathbf{x}_i) + b \leq -1, & \text{if } \; y_i = -1 \end{cases}$$

which is equivalent to

$$y_i[\mathbf{w}^T \varphi(\mathbf{x}_i) + b] \geq 1, \qquad i = 1, ..., N.$$

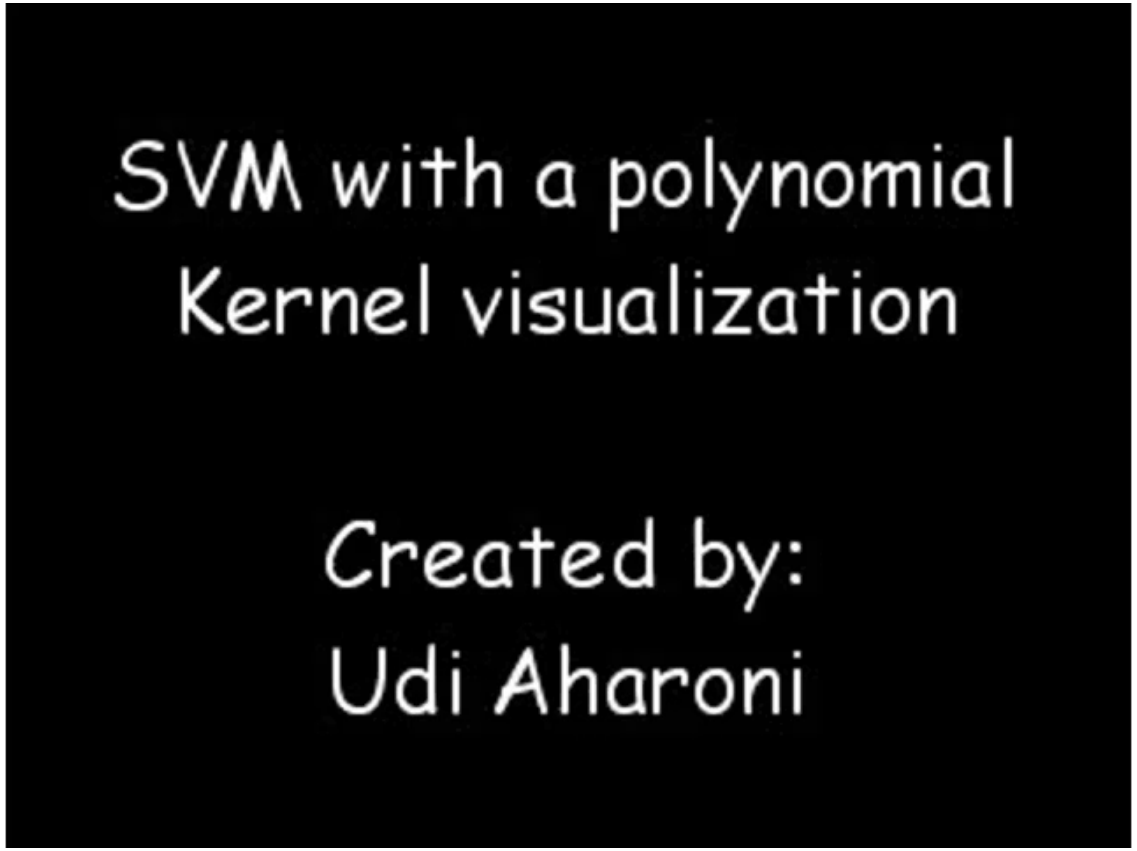
SVM Challenges

Kernels

What is the $\phi(.)$

Non-linear Separability

What if data can't be separated by straight line?



SVM

- $\varphi(\cdot)$ is a non-linear function that maps the input space to a high (possibly infinite) dimensional feature space
- Equation 5 constructs a hyperplane $\mathbf{w}^T \varphi(\mathbf{x}) + b = 0$ between the two classes
- Goal: maximize margin between both classes
  $\rightarrow$ minimizing $\mathbf{w}^T \mathbf{w}$

SVM

This convex optimization problem is defined as:

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \mathcal{J}(\mathbf{w}, b, \boldsymbol{\xi}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C \sum_{i=1}^{N} \xi_i \tag{7}$$

subject to

$$\begin{cases} y_i[\mathbf{w}^T\varphi(\mathbf{x}_i) + b] \geq 1 - \xi_i, & i = 1, ..., N \\ \xi_i \geq 0, & i = 1, ..., N. \end{cases} \tag{8}$$

- ξ_i are slack variables which are needed to allow misclassifications in the set of inequalities
- C is the regularisation coefficient $\rightarrow$ tuned

SVM Objective Function

Equation 7 Components

First term: Maximizes margin between classes in feature space while regularizing to penalize large weights

Second term: Minimizes misclassification errors

Balancing act between:

- Model complexity (regularization)
- Training accuracy (error minimization)
- Generalization performance

Kernel Trick for Non-linearity

This leads to the following classifier (Cristianini & Shawe-Taylor, 2000):

$$y(\mathbf{x}) = \text{sign}[\sum_{i=1}^{N} \alpha_i \, y_i \, K(\mathbf{x}_i, \mathbf{x}) + b], \tag{9}$$

- The Lagrange multipliers α_i are determined by optimizing the dual problem
- Low-noise problems, many of the α_i will typically be equal to zero (sparseness property)
- Support vectors: the training observations corresponding to non-zero α_i

Transform data into higher-dimensional space where linear separation becomes possible

Additional resources: Oxford ML lectures

Kernel trick

$K(\mathbf{x}_i, \mathbf{x}) = \varphi(\mathbf{x}_i)^T \varphi(\mathbf{x})$ is taken with a positive definite kernel satisfying the Mercer theorem.

The following kernel functions $K(\cdot, \cdot)$ were used:

$$K(\mathbf{x}, \mathbf{x}_i) = (1 + \mathbf{x}_i^T \mathbf{x}/c)^d, \quad \text{(polynomial kernel)}$$
$$K(\mathbf{x}, \mathbf{x}_i) = \exp\{-\|\mathbf{x} - \mathbf{x}_i\|_2^2/\sigma^2\}, \quad \text{(RBF kernel)}$$

where d, c and σ are constants.

$\rightarrow$ black-box model

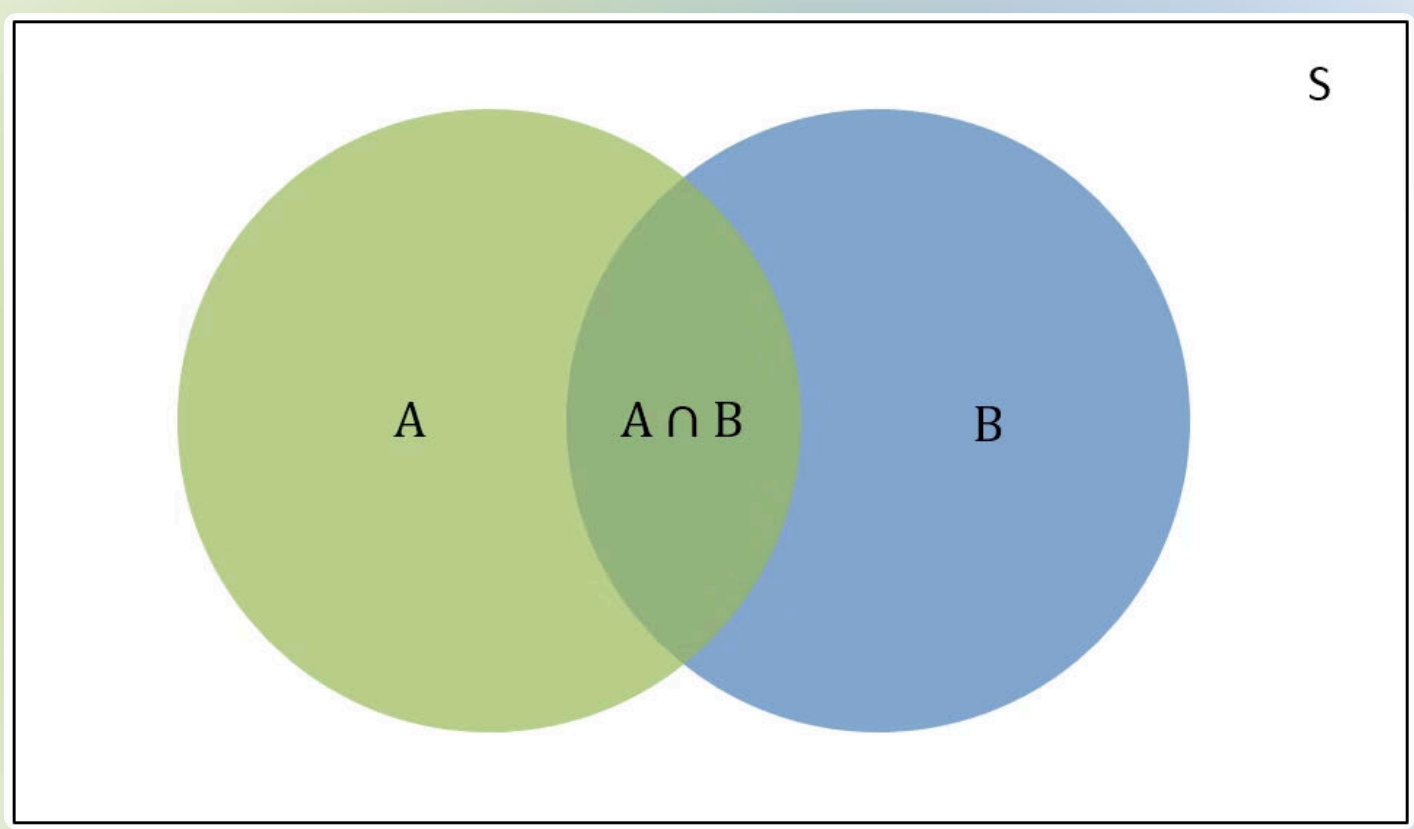
Hyperparameter Tuning: GridSearch

Systematic exploration of parameter space using cross-validation with weighted AUC by class size

1. Grid is determined by the two input parameters
2. 2-fold cross validation on the initial grid
3. 10-fold cross validation on the best point and its adjacent point
4. If a better pair is found: repeat procedure on its neighbours
5. Stop: no better pair found or border reached

Naïve Bayes Classifier

Probabilistic classification based on Bayes' theorem with independence assumptions



Bayes Theorem Recap

Conditional Probability

Probability of event A given event B has occurred

	Female	Male	Total
Teacher	8	12	20
Student	32	48	80
Total	40	60	100

$$P(A \mid B) = \frac{P(A, B)}{P(B)}$$

$$P(B \mid A) = \frac{P(A, B)}{P(A)}$$

$$P(Teacher \mid Male) = \frac{P(Teacher \bigcap Male)}{P(Male)} = 12/60 = 0.2$$

Foundation for updating beliefs with new evidence

Bayes Theorem Fundamentals

Conditional Probability

P(A|B) = likelihood of A given B occurred

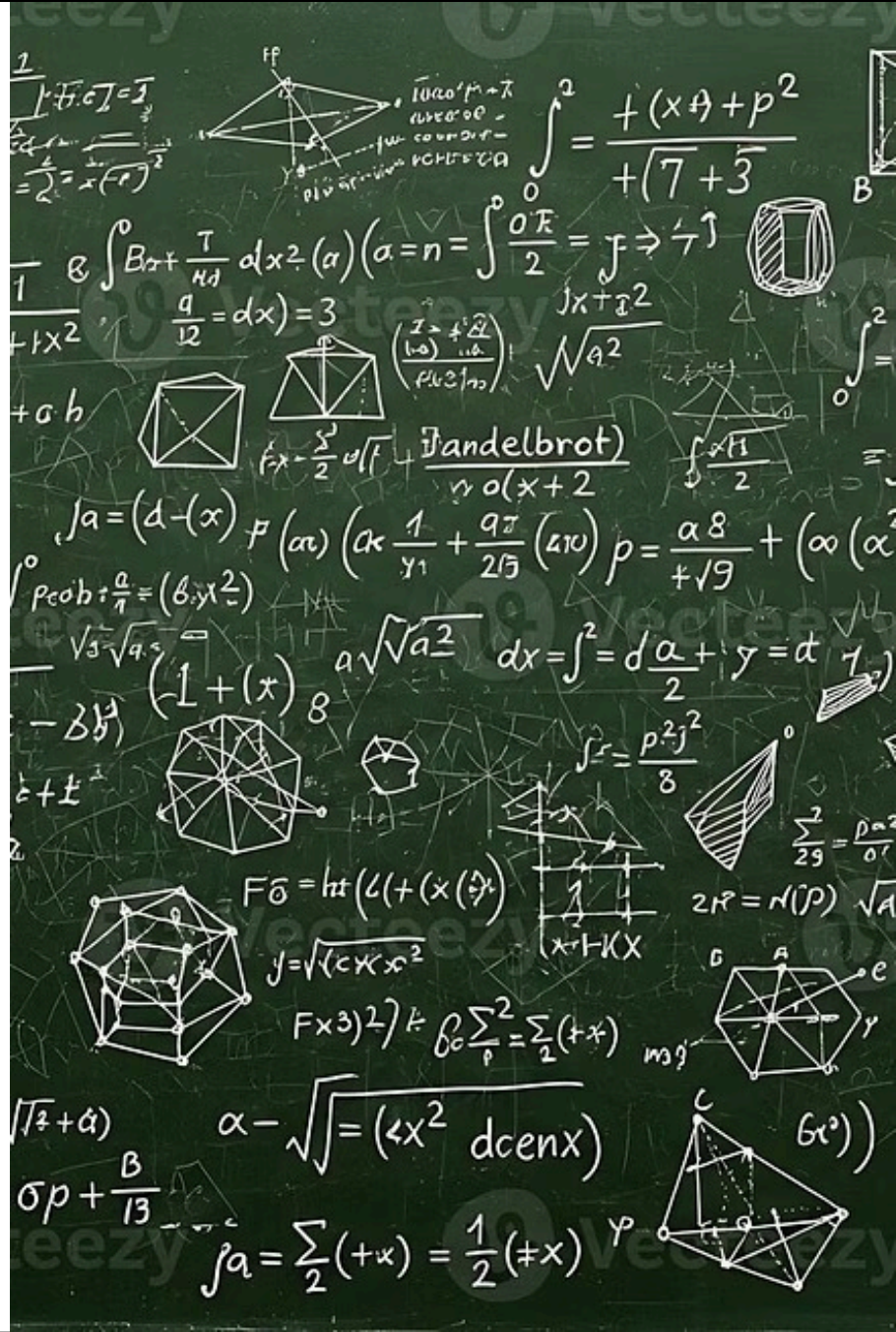
$$P(A \mid B) = \frac{P(A, B)}{P(B)}$$

$$P(B \mid A) = \frac{P(A, B)}{P(A)} \qquad P(A, B) = P(B \mid A)\, P(A)$$

Bayes Theorem

Reverses conditional probability using prior knowledge

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)}$$



Naïve Bayes with Multiple Features

In real world problem: multiple X-variables

Independence Assumption

- Features assumed conditionally independent given class label

Posterior Probability = Probability of outcome

- Combine prior with likelihood of all features

$$P(A = k|\ X_1 \dots X_n) = \frac{(P(X1|\ A{=}k){*}P(X2|\ A{=}k){*}\dots{*}P(Xn|\ A{=}k){*}P(A{=}k))}{(P(X1){*}P(X2){*}\dots{*}P(Xn))}$$

Probability of Outcome | Evidence = (Prob. of Likelihood of evidence □Prior) / (Prob. of Evidence)

—> "Naïve" because real-world features often correlate, yet method works surprisingly well

Example Dataset

Fruit Classification Data

Dataset with probabilities:

Fruit	Long (x1)	Sweet (x2)	Yellow (x3)
Orange	0	1	0
Banana	1	0	1
Banana	1	1	1
Other	1	1	0
..	..	..	..

Aggregated to a counts table:

Type	Long	Not Long	Sweet	Not Sweet	Yellow	Not Yellow	**Total**
Banana	400	100	350	150	450	50	500
Orange	0	300	150	150	300	0	300
Other	100	100	150	50	50	150	200
Total	500	500	650	350	800	200	1000

Source: Machine Learning Plus

Naive Bayes example

- Objective: predict if a given fruit is Banana, Orange, or Other when only three features are known (long, sweet and yellow)
- Step 1: compute Prior probabilities for each class of fruits:
  - P(Y = Banana) = 500/1000 = 0.50
  - P(Y = Orange) = 300/1000 = 0.30
  - P(Y = Other) = 200/1000 = 0.20

Type	Long	Not Long	Sweet	Not Sweet	Yellow	Not Yellow	**Total**
Banana	400	100	350	150	450	50	500
Orange	0	300	150	150	300	0	300
Other	100	100	150	50	50	150	200
Total	500	500	650	350	800	200	1000

Naive Bayes example

Step 2: Compute the probability of evidence that goes in the denominator.

- P(X1 = Long) = 500/1000 = 0.50
- P(X2 = Sweet) = 650/1000 = 0.65
- P(x3 = Yellow) = 800/1000 = 0.80

Type	Long	Not Long	Sweet	Not Sweet	Yellow	Not Yellow	**Total**
Banana	400	100	350	150	450	50	500
Orange	0	300	150	150	300	0	300
Other	100	100	150	50	50	150	200
Total	500	500	650	350	800	200	1000

Naive Bayes example

Step 3: Compute the probability of likelihood of evidences that goes in the numerator.

- P(X1 = Long | Y = Banana) = 400/500 = 0.80
- P(X2 = Sweet | Y = Banana) = 350/500 = 0.70
- P(X3 = Yellow | Y = Banana) = 450/500 = 0.90

Type	Long	Not Long	Sweet	Not Sweet	Yellow	Not Yellow	**Total**
Banana	400	100	350	150	450	50	500
Orange	0	300	150	150	300	0	300
Other	100	100	150	50	50	150	200
Total	500	500	650	350	800	200	1000

Combined likelihood: 0.8 × 0.7 × 0.9 = 0.504

=> The overall probability of Likelihood of evidence for Banana

Naive Bayes example

FinoStep 4: Substitute all the 3 equations into the Naive Bayes formula, to get the probability that it is a banana.

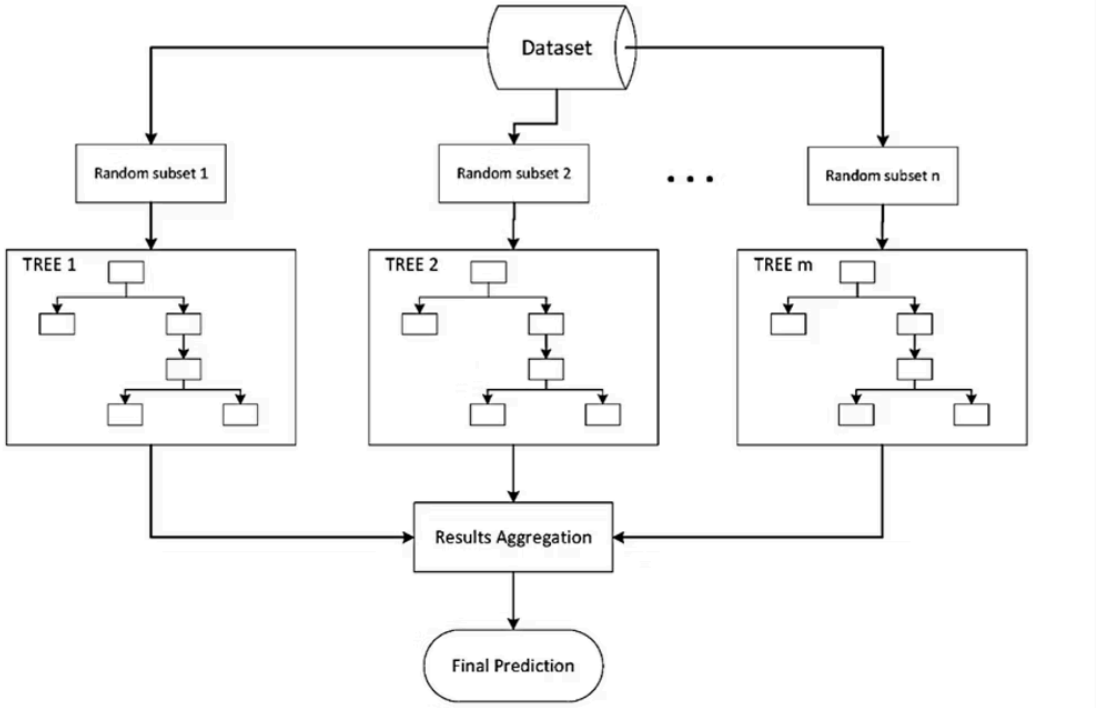
- P(**Banana** | Long, Sweet, Yellow) = **0.252** | P(Evidence). ß = 0.504 * 0.5
- P(Orange | Long, Sweet, Yellow) = 0 | P(Evidence)
- P(Other | Long, Sweet, Yellow) = 0.01875| P(Evidence)

Type	Long	Not Long	Sweet	Not Sweet	Yellow	Not Yellow	Total
Banana	400	100	350	150	450	50	500
Orange	0	300	150	150	300	0	300
Other	100	100	150	50	50	150	200
Total	500	500	650	350	800	200	1000

P(evidence) is the same for all, so optional to compute

Laplace Correction

- The value of P(Orange | Long, Sweet and Yellow) was zero in the above example, because P(Long | Orange) was zero.
- When you have a model with many features, the entire probability will become zero because one of the feature's value was zero.
- To avoid this, we increase the count of the variable with zero to a small value (usually 1) in the numerator, so that the overall probability doesn't become zero.



Ensemble Methods

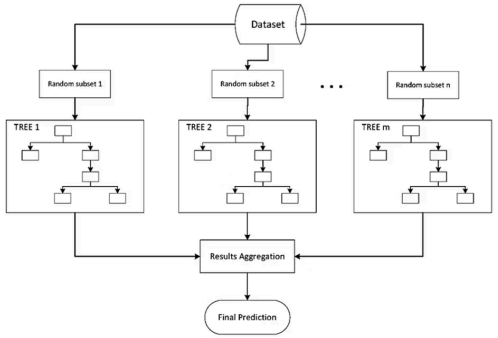
Combining multiple classifiers for superior predictive performance

Core Principle

Train multiple models from data

Prediction Strategy

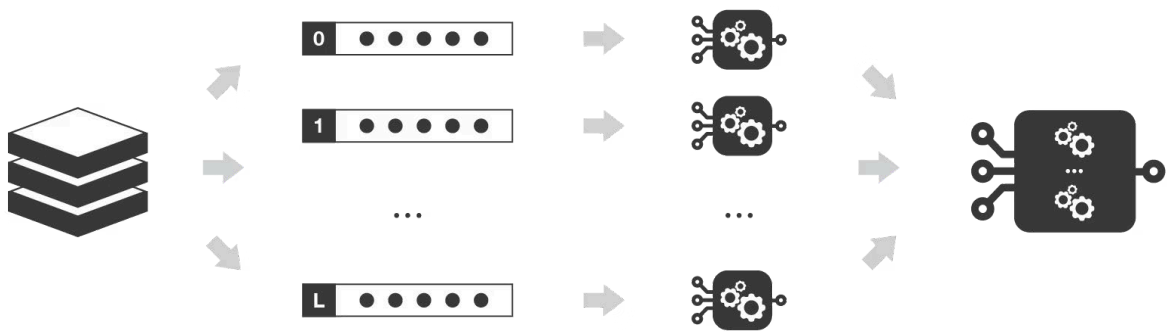
Combine classifier outputs for final decision



Ensemble Method Categories

1	2	3
Bagging	**Boosting**	**Stacking**
Homogeneous weak learners trained independently in parallel, combined via deterministic averaging	Homogeneous weak learners trained sequentially—each model adapts based on predecessors, combined deterministically	Heterogeneous weak learners trained in parallel, meta-model learns to combine their predictions

Bagging: Bootstrap Aggregating



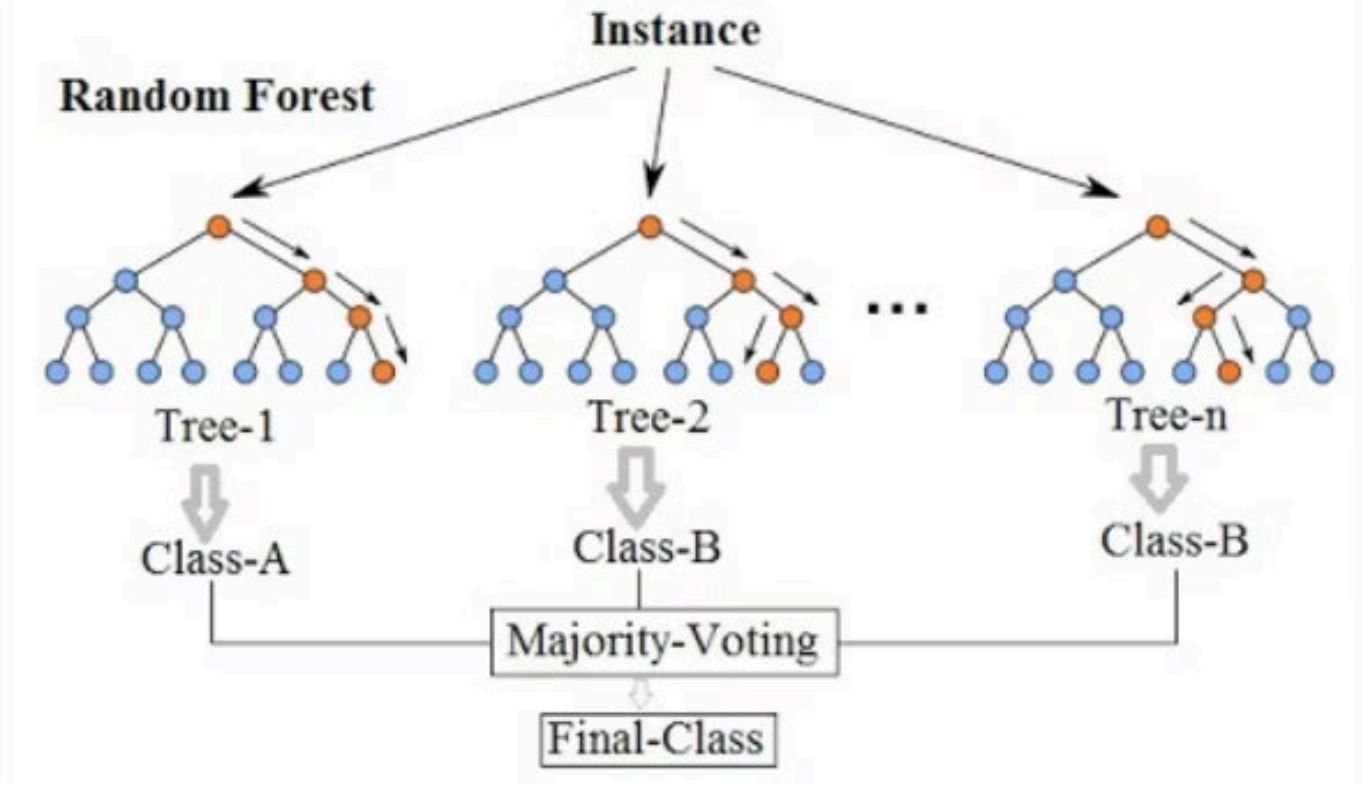
initial dataset | L bootstrap samples | weak learners fitted on each bootstrap sample | ensemble model (kind of average of the weak learners)

Random sampling with replacement creates diverse training sets

Reduces variance, improves stability

Deep dive: Ensemble methods

Random Forest

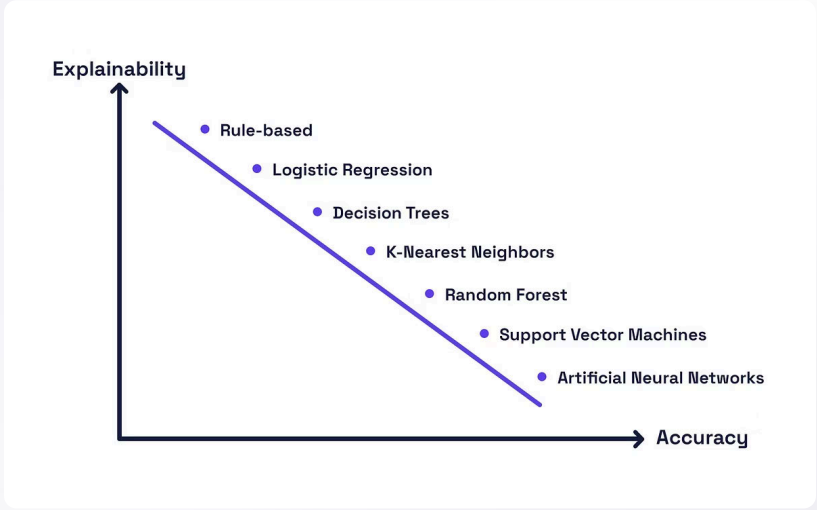


Specialized Bagging

Ensemble of decision trees with dual randomization:

- Random data sampling (bagging)
- Random feature sampling per split
- Trick: also samples over features!
- Majority voting for final prediction

Image source



Decision Trees vs Random Forests

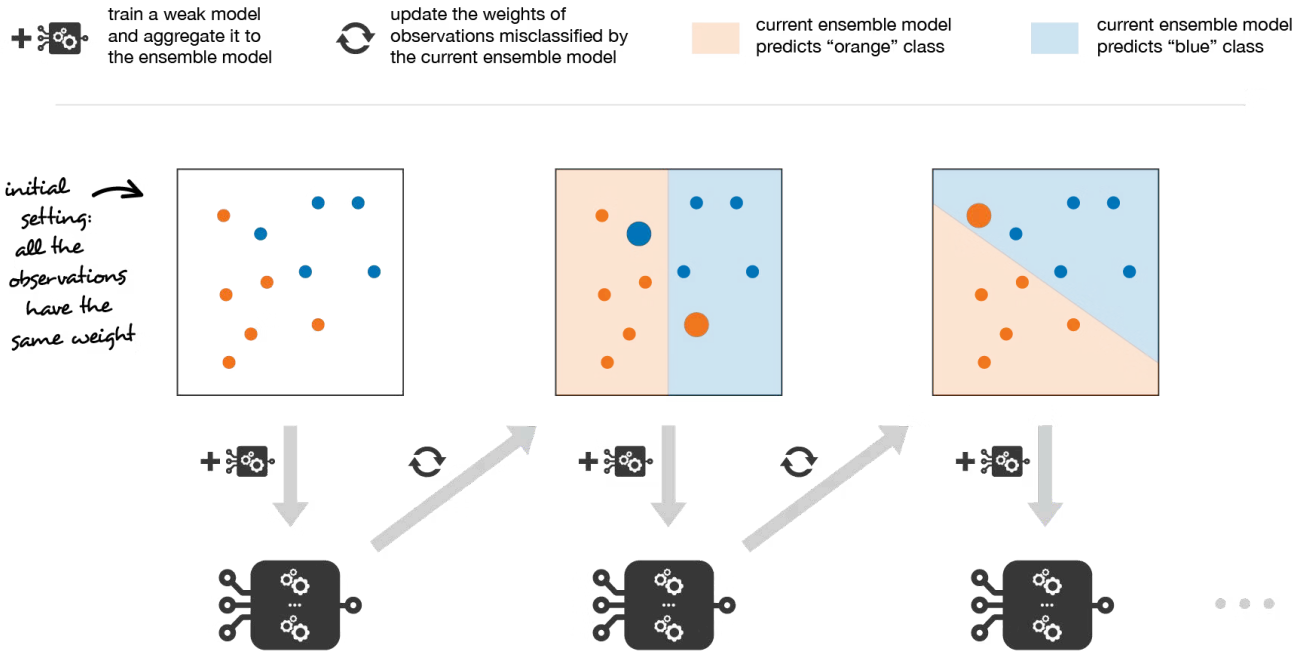
Decision Trees

- Fast construction
- Interpretable results (small trees)
- Prone to overfitting without pruning

Random Forests

- Multiple decision trees
  - Using different data sample (bagging) and random training features
- Prevents overfitting naturally
- Higher accuracy, reduced variance

Boosting: Sequential Learning



Adaptive Weight Adjustment

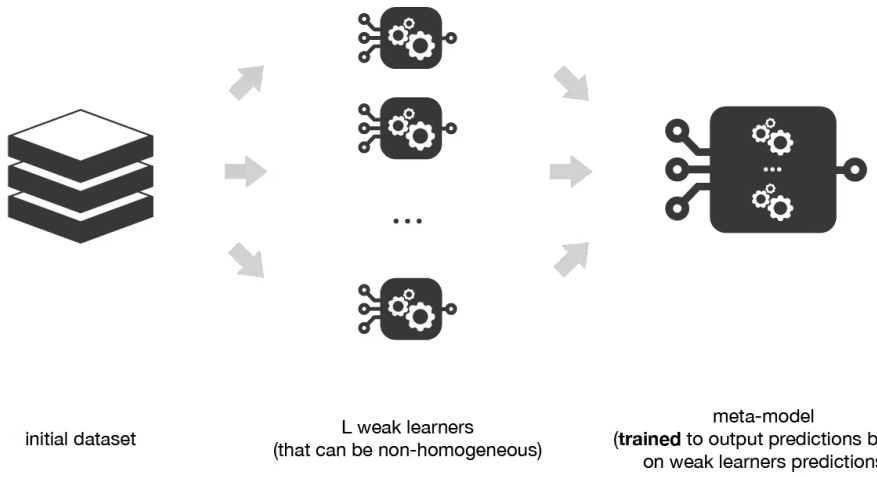
Iterative training where each model focuses on previous errors

- Increase weights for misclassified records
- Decrease weights for correct predictions

Popular Techniques

- **Gradient Boosting**
- **AdaBoost (figure)**

Stacking: Heterogeneous Learning



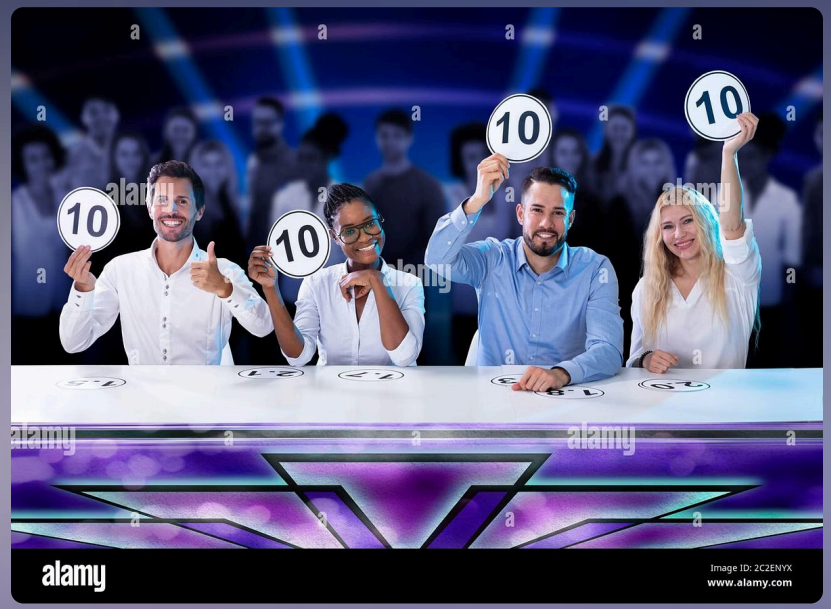
Meta-Model Architecture

Learn several different weak learners and combine them by training a meta-model to output predictions based on the multiple predictions returned by these weak models.

Example weak learner stack:

1. K-NN classifier
2. Logistic regression
3. Support vector machine

=> Neural network learns optimal combination of base predictions



Model Evaluation

Measuring and validating classification performance

Training and Test Splits

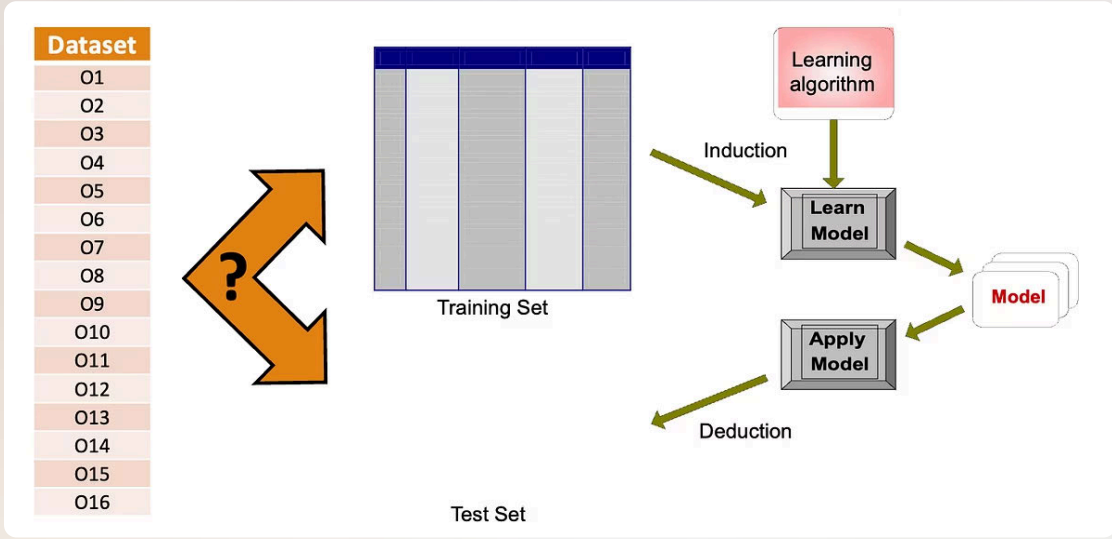
Critical question: How should observations be allocated between training and test sets?

Random selection?

Stratified sampling?

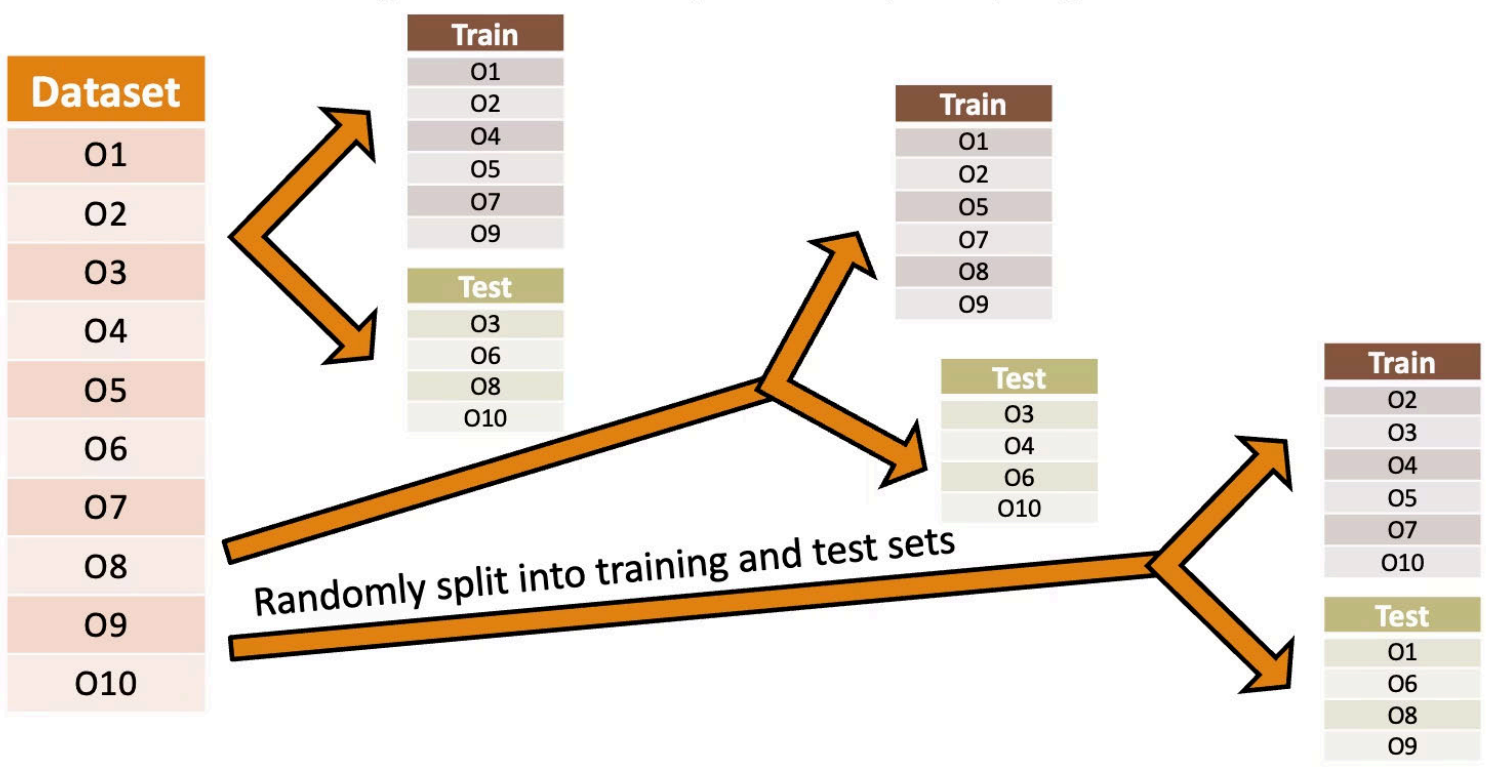
Sequential partitioning?

Cross-validation folds?

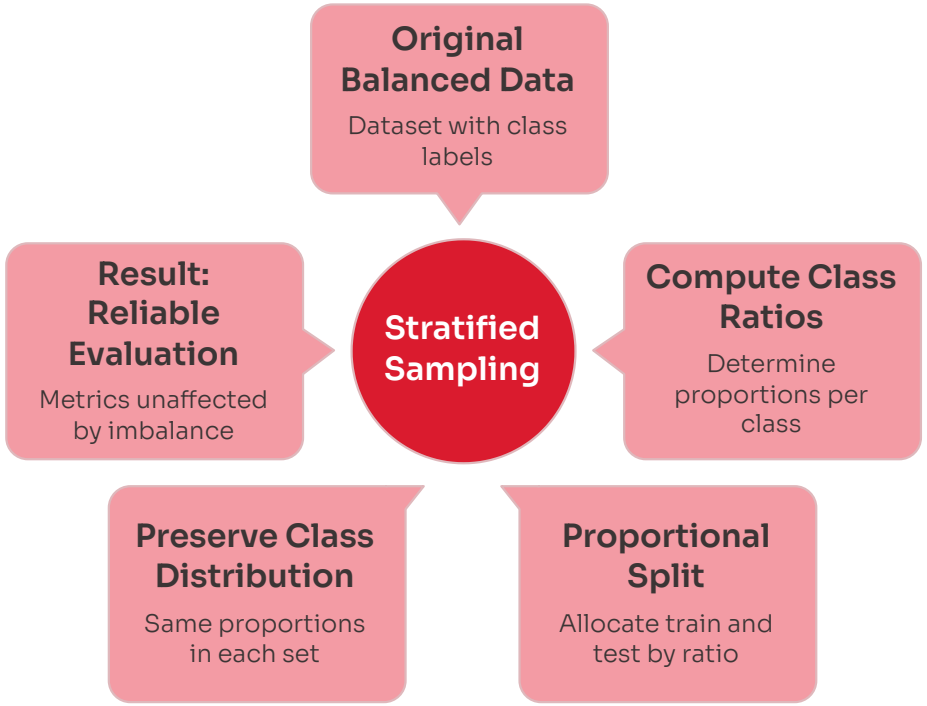


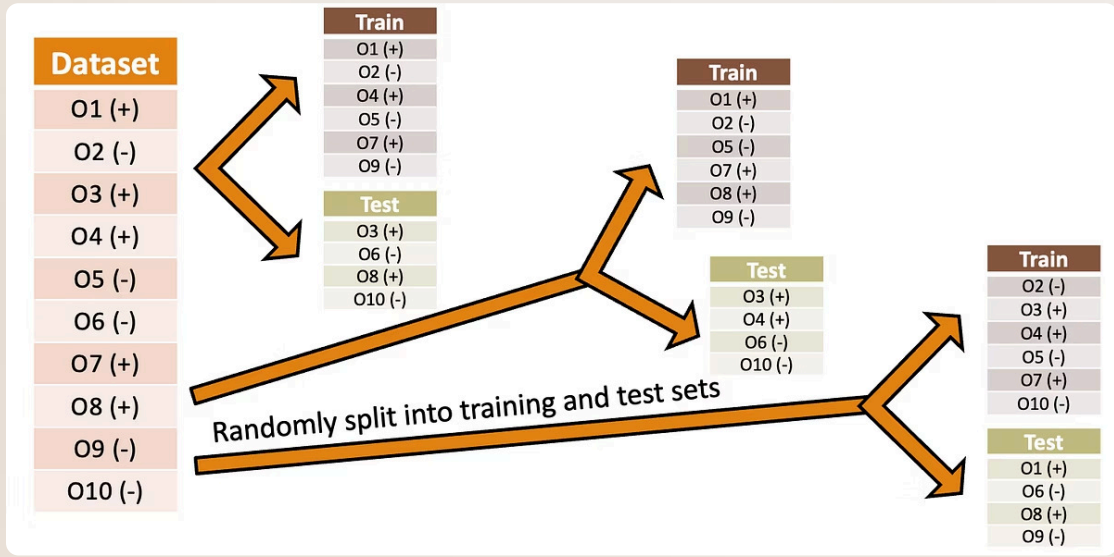
Random Resampling

Generate **multiple train-test splits through** random sampling—ensures diverse evaluation across data variations

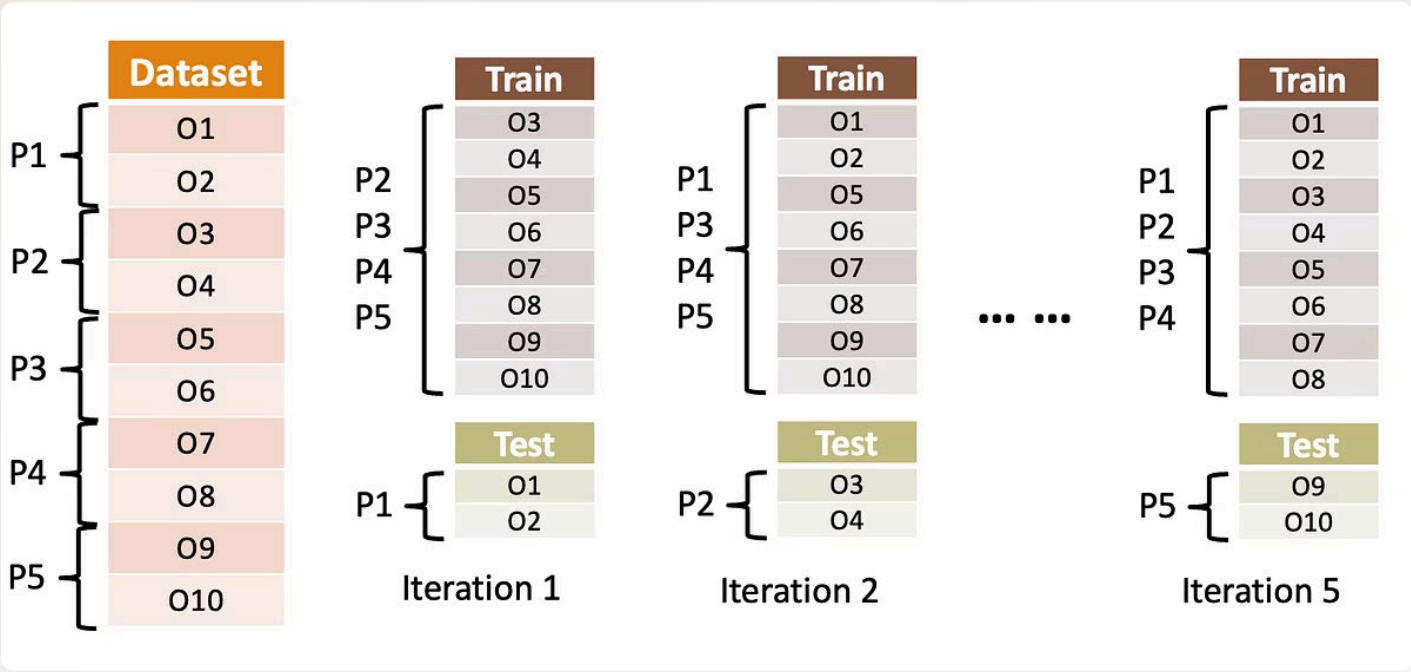


Stratified Sampling

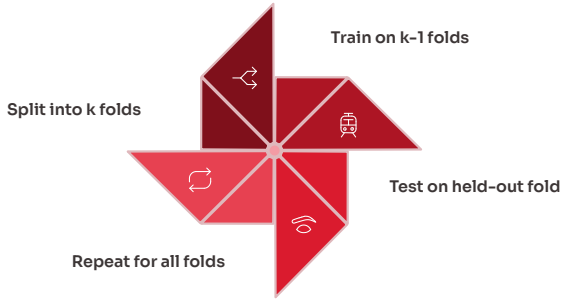




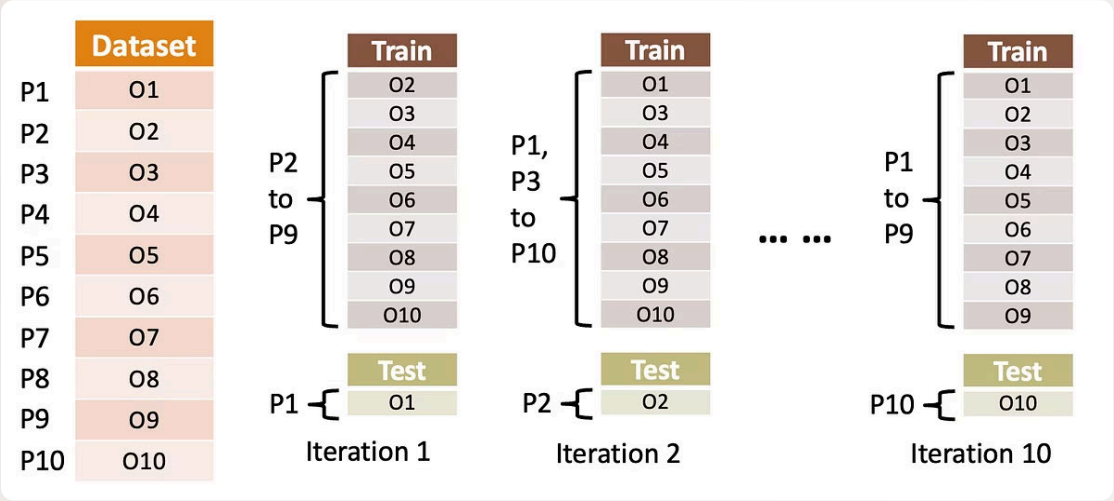
Maintains class label proportions in both training and test sets—critical for imbalanced datasets



K-Fold Cross Validation



Systematic rotation through *k* partitions maximizes data utilization for both training and evaluation

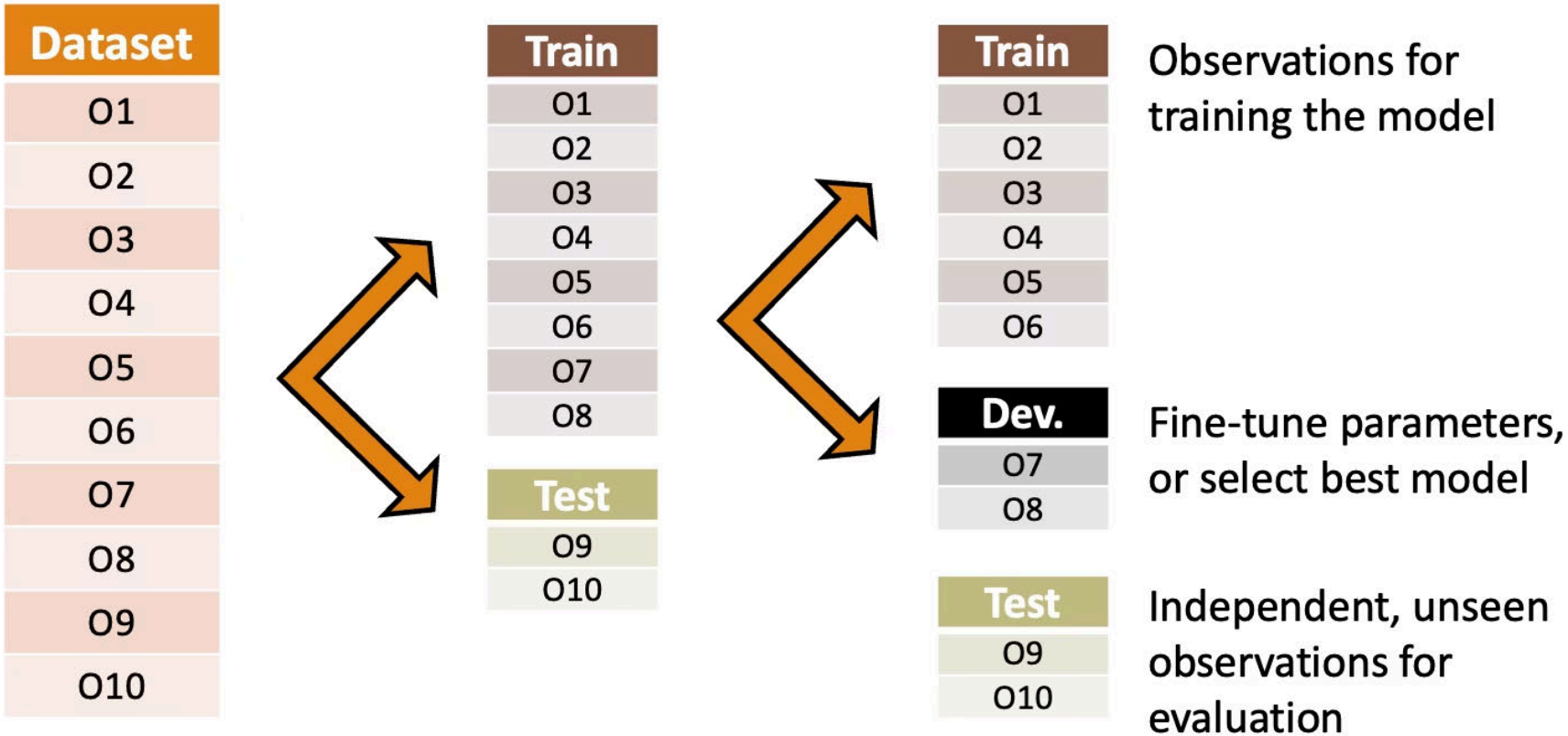


Leave-One-Out Validation

Extreme case where *k* equals number of observations —computationally intensive but maximally unbiased

Development Set (Validation)

Separate validation set prevents overfitting during model selection and hyperparameter optimization



Confusion Matrix

Understanding Predictions

Visual representation of classification results

Example applications:

- Medical diagnosis (disease detection)
- Information retrieval (relevance determination)
- Quality control (defect identification)

	Ground Truth: Yes	**Ground Truth: No**
Predicted: Yes	True Positive (TP)	False Positive (FP)
Predicted: No	False Negative (FN)	True Negative (TN)

Medical Testing Example

Disease Detection Results

- **TP = 80:** Correctly identified patients with disease
- **FN = 20:** Missed diagnoses (disease present, test negative)
- **FP = 10:** False alarms (healthy patients flagged)
- **TN = 90:** Correctly identified healthy patients

	Predicted Positive (Disease)	Predicted Negative (No Disease)
Actual Positive (Disease)	**TP = 80**	**FN = 20**
Actual Negative (No Disease)	**FP = 10**	**TN = 90**

Total: 200 patients tested

Accuracy Metric

Definition

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

Percentage of correct classifications across all observations

Limitation

Imbalanced classes problem: High accuracy can be misleading when one class dominates

Example: 95% accuracy predicting "No disease" when 95% of patients are healthy is uninformative

Precision and Recall

Precision

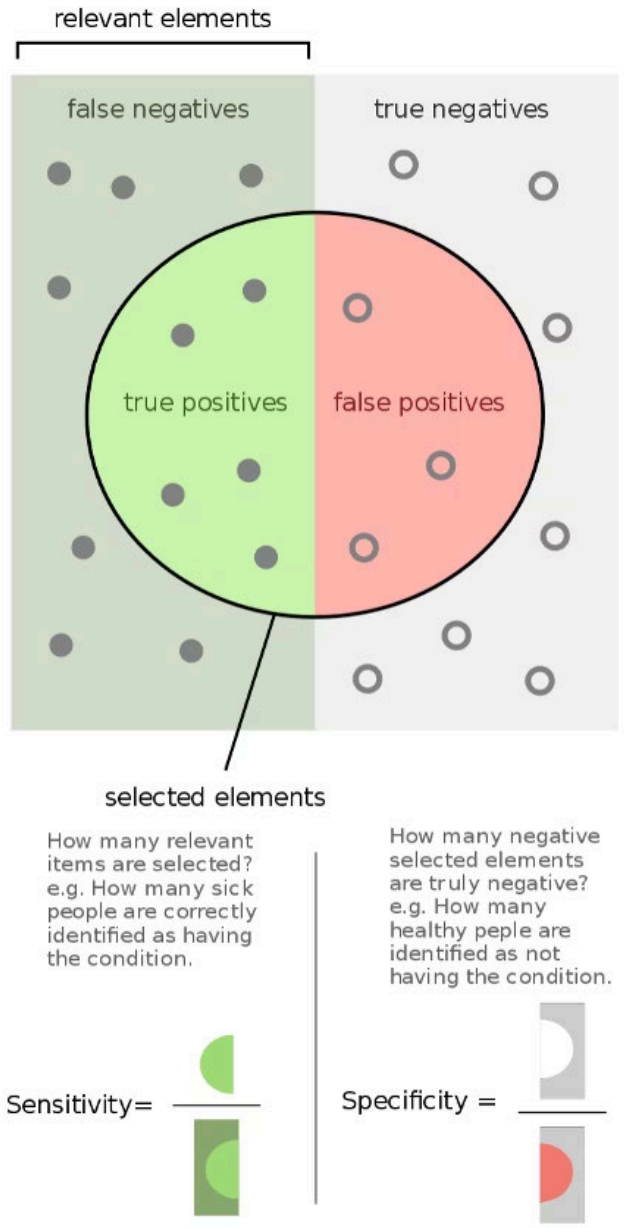
$$\text{Precision} = \frac{TP}{TP + FP}$$

Of all items classified positive, what percentage are truly positive?

Recall (Sensitivity)

$$\text{Recall} = \frac{TP}{TP + FN}$$

Of all truly positive items, what percentage did we identify?



Specificity

Definition

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

Of all truly negative cases, what percentage are correctly identified?

		Ground Truth (Relevant)	
		Yes	**No**
Classification (Retrieved)	**Yes**	TP	FP
	No	FN	TN

Interpretation

Measures model's ability to identify negatives correctly

High specificity = few false alarms

Complements sensitivity for complete picture

F1-score

The Challenge: Precision & Recall Trade-off

Precision and Recall often present an inverse relationship: improving one tends to decrease the other. For example, a model can achieve **100% recall** by simply classifying every instance as positive, but this would dramatically lower its precision due to many false positives.

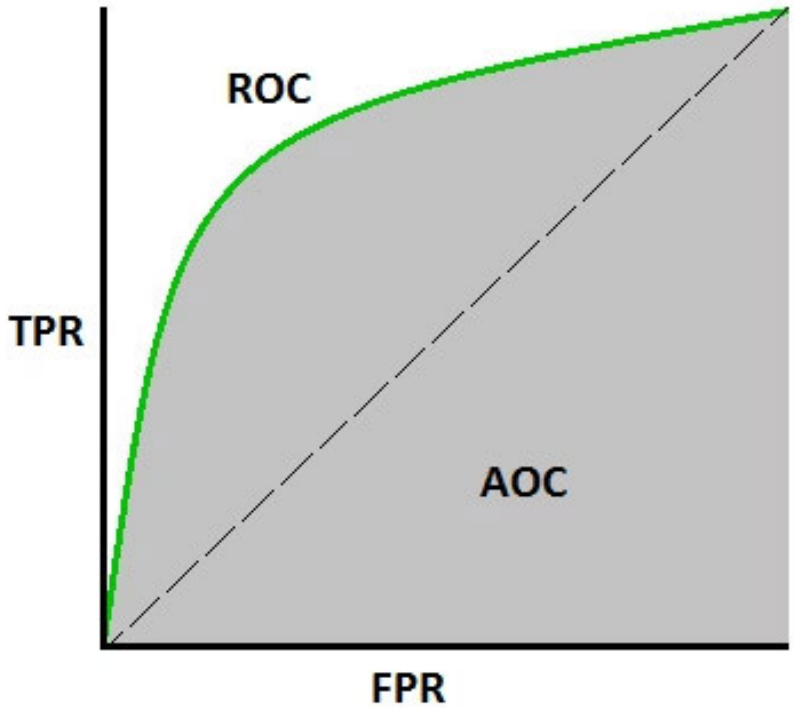
The F1-score addresses this by providing a single metric that balances both concerns.

F1-score: The Harmonic Mean

$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

The F1-score is the **harmonic mean** of Precision and Recall. It gives equal weight to both metrics, making it particularly useful for evaluating models on **imbalanced datasets** where a simple accuracy score can be misleading.

ROC Curve Analysis



Receiver Operating Characteristic

Plots True Positive Rate (y-axis) vs False Positive Rate (x-axis)

- Shows classification performance across thresholds
- Diagonal line = random classifier
- AUC (Area Under Curve): 0.5 = random, 1.0 = perfect
- Higher AUC indicates better model discrimination

Case Study: Composer Classification

Research Application

Classifying MIDI files by composer using machine learning models

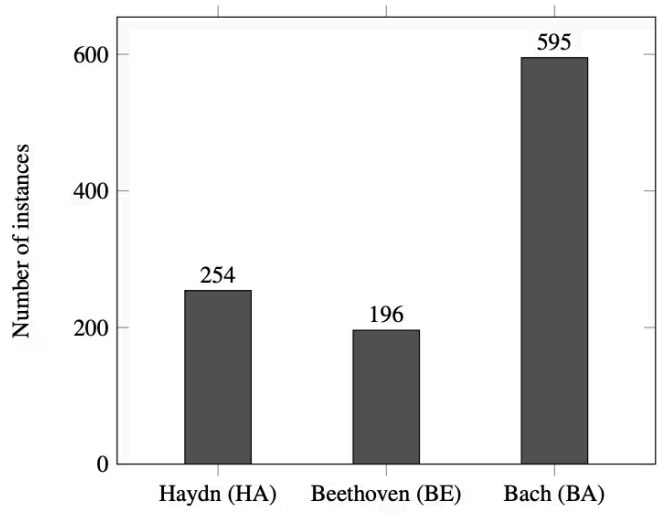


Table 2 Performance of the models with 10-fold cross-validation

Method	Accuracy	AUC
C4.5 Decision tree	*80%*	*79%*
RIPPER ruleset	*81%*	*85%*
Logistic regression	*83%*	92%
Naive Bayes	*80%*	90%
Support vector machines	**86%**	**93%**

$p < 0.01$: italic, $p > 0.05$: bold, best: bold.

Table 9 Confusion matrix for support vector machines

a	b	c	classified as
204	26	24	a = HA
49	**127**	20	b = BE
22	10	**563**	c = BA

Herremans et al. 2015, Computational Music Analysis

Knowledge Check

Test Your Understanding

Complete the quiz to assess your grasp of classification algorithms and evaluation metrics

Take Quiz