UNIVERSITÄT LEIPZIG
Medizinische Fakultät

Universitätsklinikum Leipzig
Medizin ist unsere Berufung.

Machine Learning

Introduction to Machine Learning

Applications in Healthcare

ScaDS.AI DRESDEN LEIPZIG

Data Science and AI for Medicine - Training School 2025

MDS Medical Data Science

Leipzig, 24 September 2025

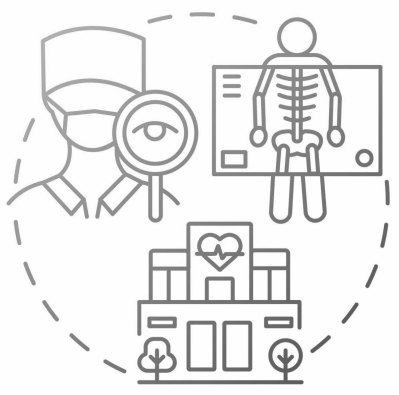
Sina Sadeghi


MACHINE LEARNING
AI

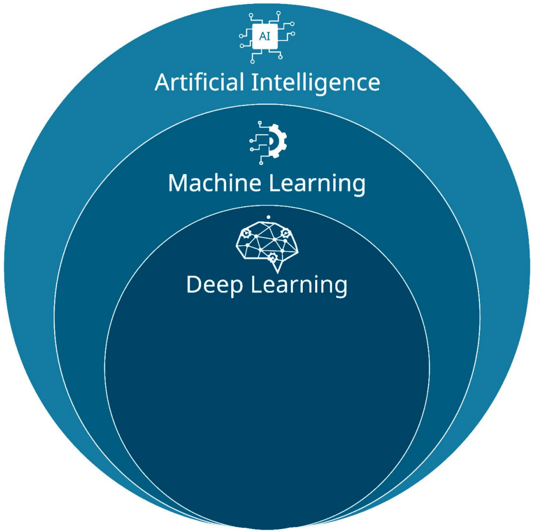
Applications

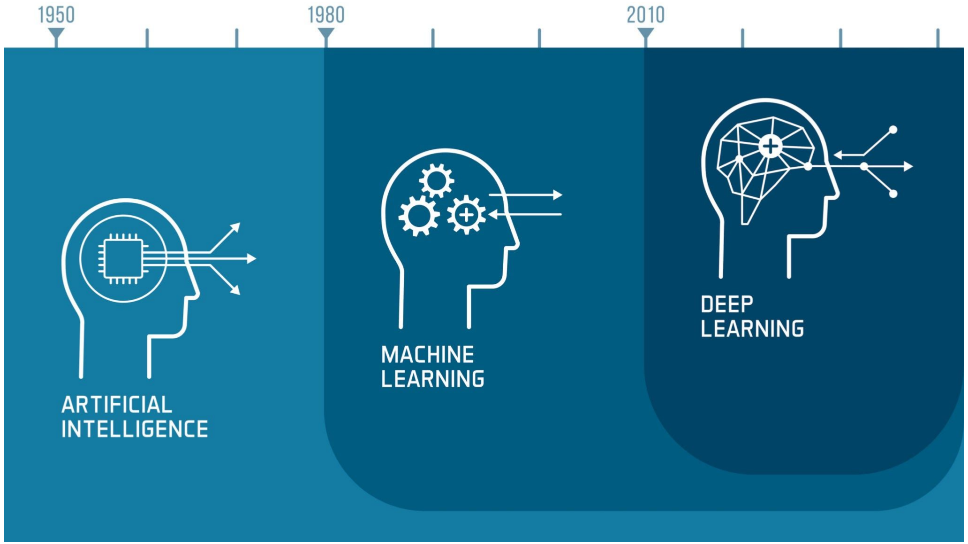
Applications in Medicine & Healthcare

- **Diagnosis & Risk Prediction**: Detect diseases from images, labs, or genomics
- **Prognosis & Survival Analysis**: Predict disease progression, hospital stay
- **Treatment Recommendation**: Personalized medicine and drug dosing
- **Medical Imaging**: Detect tumors, segment organs, assist radiologists
- **Resource Optimization**: Predict patient flow, optimize hospital resources



Machine Learning vs Deep Learning & Artificial Intelligence



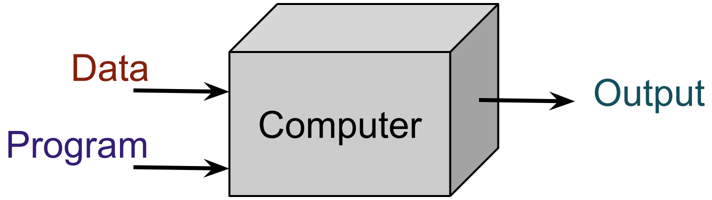


AI: techniques that enables computers to mimic human intelligence / behaviour

ML: ability of computers to learn without explicitly being programmed

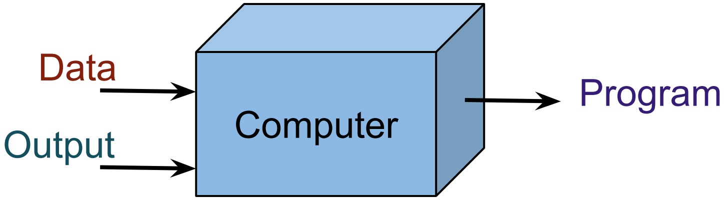
DL: learning complex patterns from DATA using multi-layered artificial neural networks

Traditional Programming



- Programmer explicitly defines logic (if–then–else)
- Works well when rules are clear and fixed
- Hand-code rules to detect diabetes from thresholds (e.g., if glucose > 140 → diabetic)

Machine Learning



- Algorithm learns patterns from examples
- Works well when rules are complex or unknown
- Train a model on patient data to learn patterns automatically

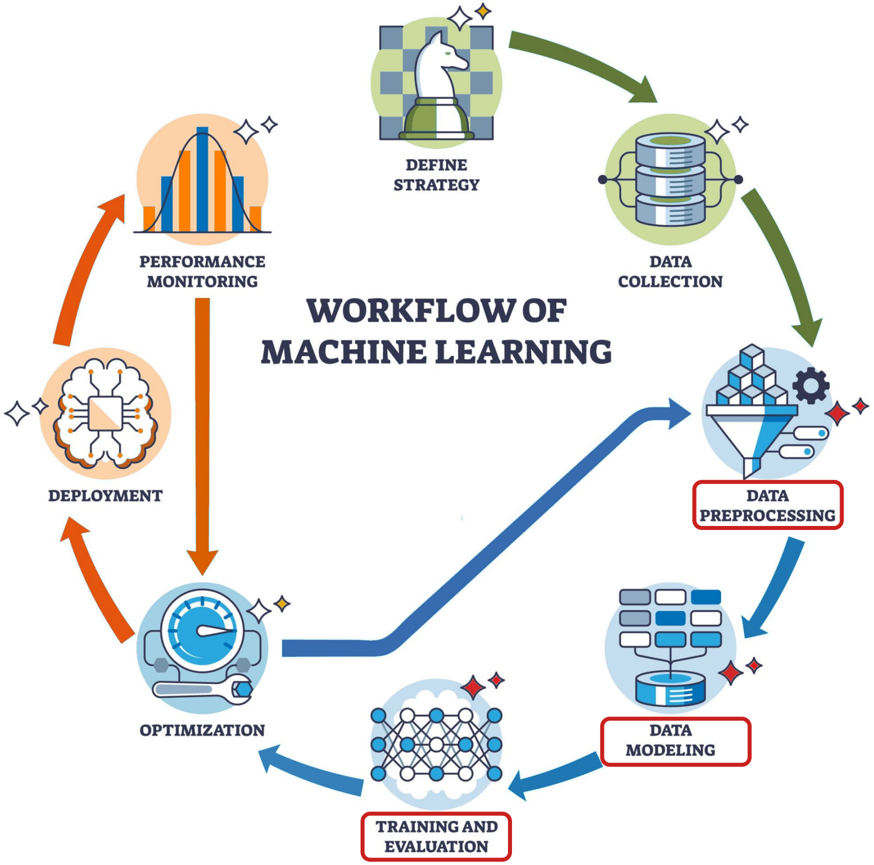
Definition

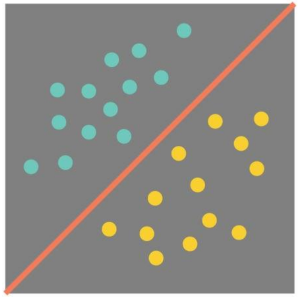
Arthur Samuel (1959): Machine Learning is a field of study that gives computers the ability to learn without being explicitly programmed.

Tom Mitchell (1998): Well-posed Learning Problem – A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.

Define Machine Learning Problem

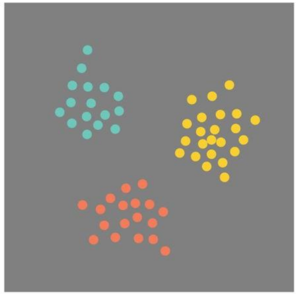
- ➔ Collect Data
- ➔ **Preprocess Data**
- ➔ **Split Data** (Train / Test / Validation)
- ➔ **Select Model**
- ➔ **Train Model**
- ➔ **Tune Hyperparameters**
- ➔ **Evaluate Model**





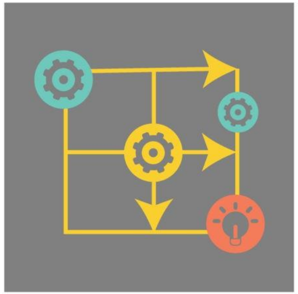
Supervised Learning

- Learns from **labeled data** (input → output)
- Goal: predict outcomes for new inputs
- Examples: classification, regression



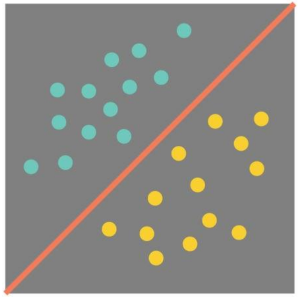
Unsupervised Learning

- Learns patterns from **unlabeled data**
- Goal: discover structure or groupings
- Examples: clustering, dimensionality reduction



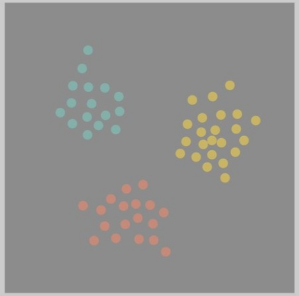
Reinforcement Learning

- Learns through interaction with an environment
- Goal: maximize cumulative reward via trial and error
- Examples: game playing, robotics



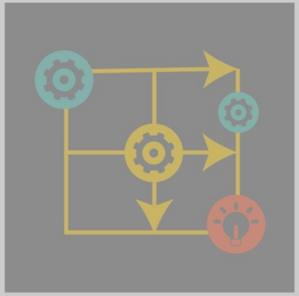
Supervised Learning

- Learns from **labeled data** (input → output)
- Goal: predict outcomes for new inputs
- Examples: classification, regression



Unsupervised Learning

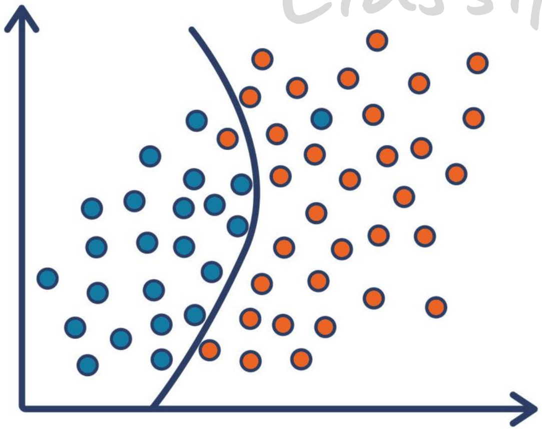
- Learns patterns from **unlabeled data**
- Goal: discover structure or groupings
- Examples: clustering, dimensionality reduction



Reinforcement Learning

- Learns through interaction with an environment
- Goal: maximize cumulative reward via trial and error
- Examples: game playing, robotics

Classification vs Regression

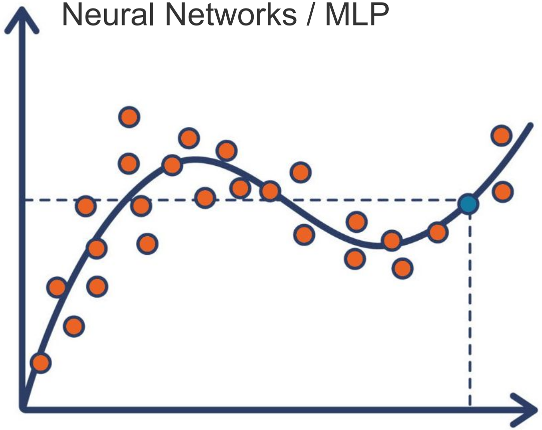


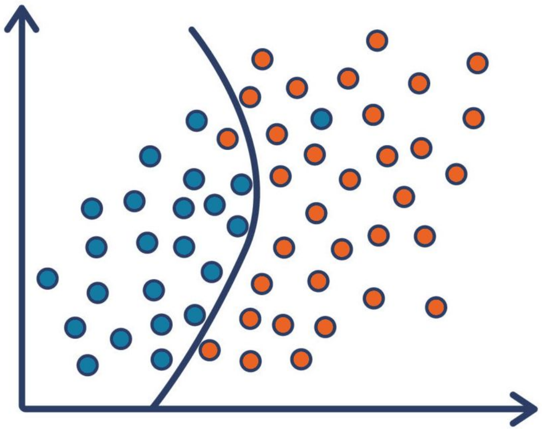
Classification

- ★ Predicts **discrete categories**
- ★ Output: class labels
  e.g., disease diagnosis: disease/no disease,
  classify tumors: benign vs. malignant
- ★ Examples: Logistic Regression, Decision Trees

Regression

- ★ Predicts **continuous values**
- ★ Output: numerical values
  e.g., patient survival time (after diagnosis/surgery),
  hospital length of stay (for admitted patients)
- ★ Examples: Linear Regression,
  Neural Networks / MLP



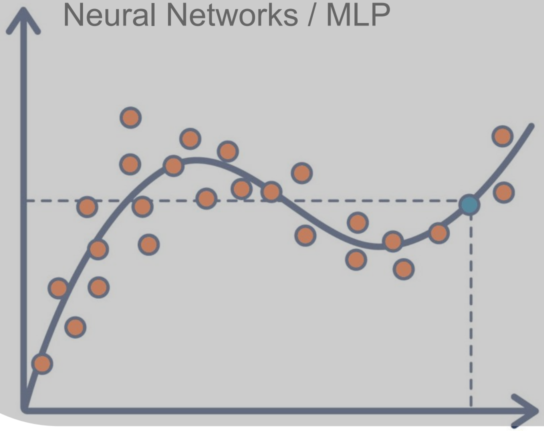


Classification

- ★ Predicts **discrete categories**
- ★ Output: class labels
  e.g., disease diagnosis: disease/no disease,
  classify tumors: benign vs. malignant
- ★ Examples: Logistic Regression, Decision Trees

Regression

- ★ Predicts **continuous values**
- ★ Output: numerical values
  e.g., patient survival time (after diagnosis/surgery),
  hospital length of stay (for admitted patients)
- ★ Examples: Linear Regression,
  Neural Networks / MLP



Dataset

Pima Indians Diabetes Database

National Institute of Diabetes and Digestive and Kidney Diseases

feature 1 feature 2

$X_i \qquad i = 1 \dots N$

ground truth

y

	Age	BMI	BloodPressure	Glucose	Insulin	SkinThickness	Pregnancies	DiabetesPedigree	Outcome
patient 1 → **0**	50	33.6	72	148	0	35	6	0.627	1
patient 2 → **1**	31	26.6	66	85	0	29	1	0.351	0
2	32	23.3	64	183	0	0	8	0.672	1
3	21	28.1	66	89	94	23	1	0.167	0
4	33	43.1	40	137	168	35	0	2.288	1

$\text{patient}_j \qquad j = 1 \dots M$

Pima Indians Diabetes Database

Using the ADAP Learning Algorithm to Forecast the Onset of Diabetes Mellitus - PMC

Example

Binary Outcome

X_1 X_2 X_i $i = 1 \dots N$

y 0/1

	Age	BMI	BloodPressure	Glucose	Insulin	SkinThickness	Pregnancies	DiabetesPedigree	Outcome
0	50	33.6	72	148	0	35	6	0.627	1
1	31	26.6	66	85	0	29	1	0.351	0
2	32	23.3	64	183	0	0	8	0.672	1
3	21	28.1	66	89	94	23	1	0.167	0
4	33	43.1	40	137	168	35	0	2.288	1

patient 1 →

patient 2 →

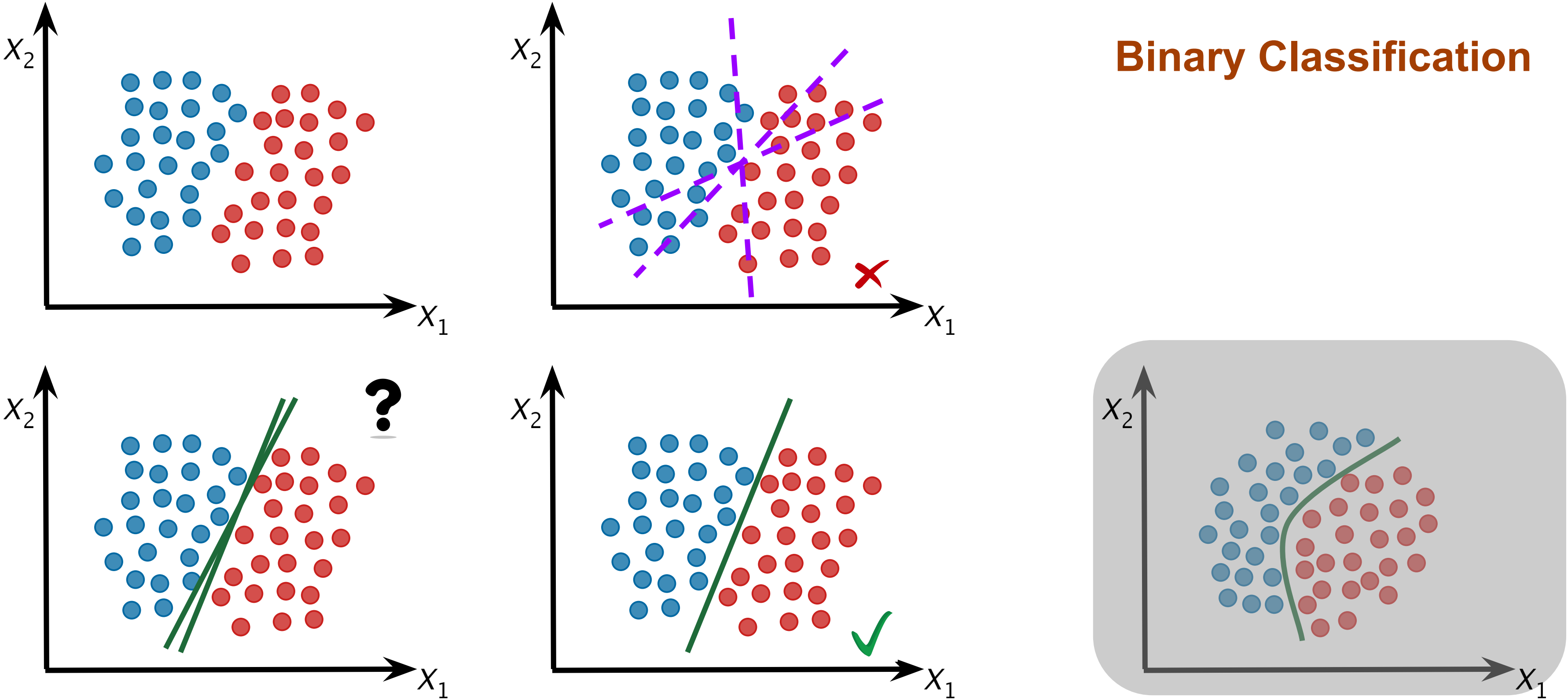
patient_j $j = 1 \dots M$

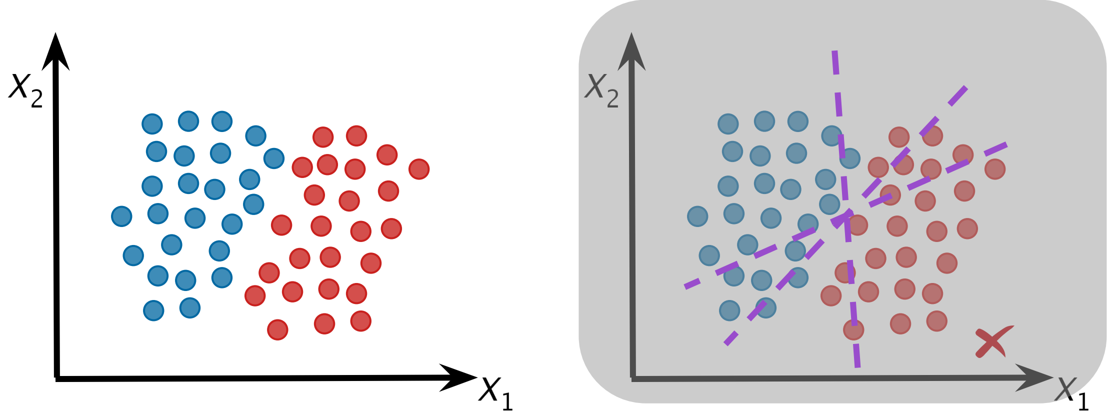
Pima Indians Diabetes Database

Using the ADAP Learning Algorithm to Forecast the Onset of Diabetes Mellitus - PMC

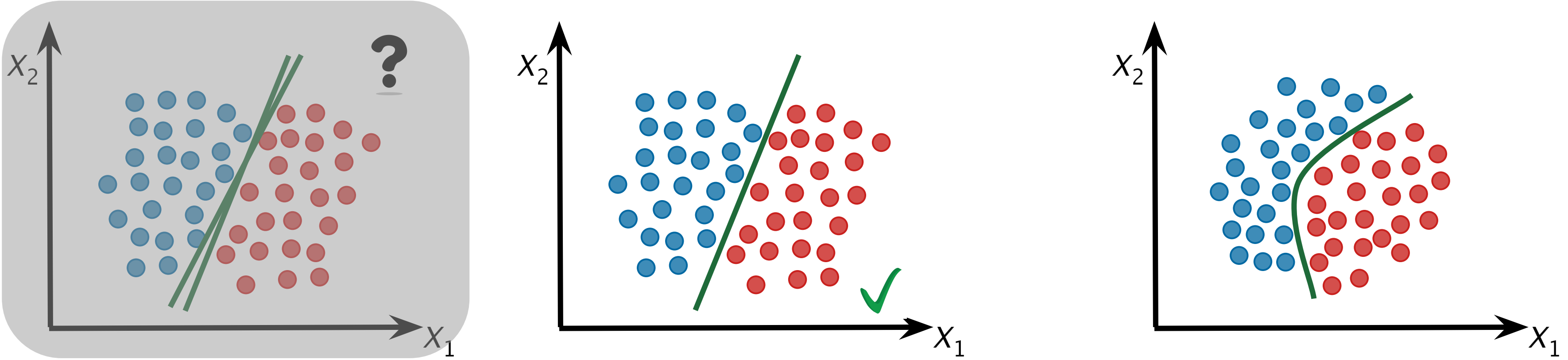
Classification

Binary Classification

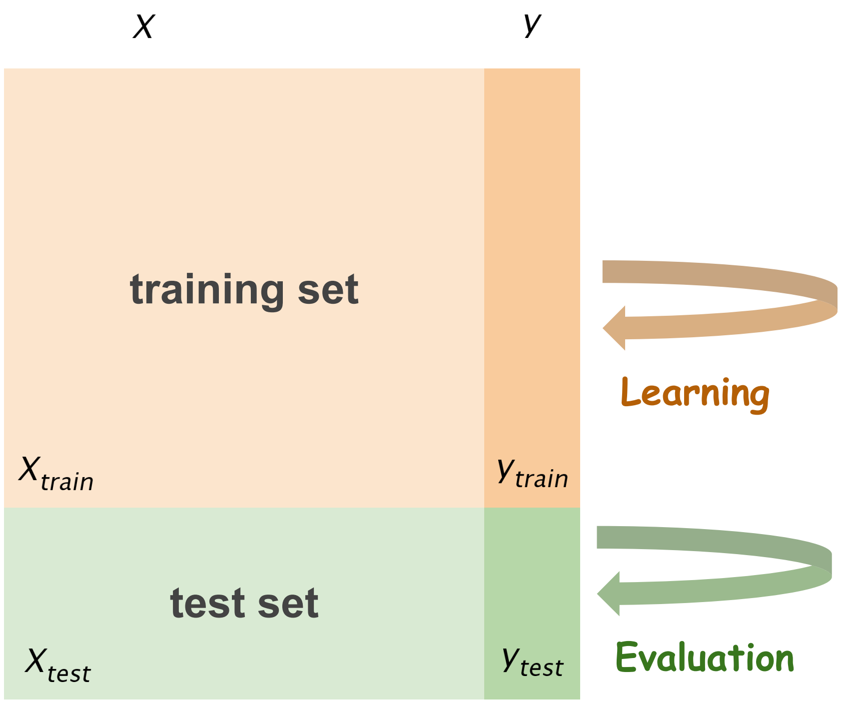




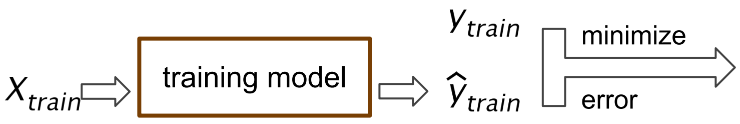
Binary Classification



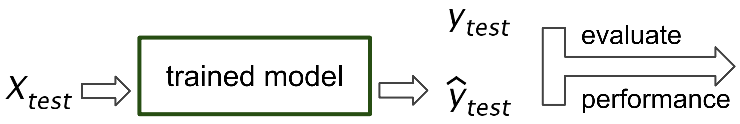
training vs. test



Dividing data into **training** and **test** sets



model learns patterns from the training set

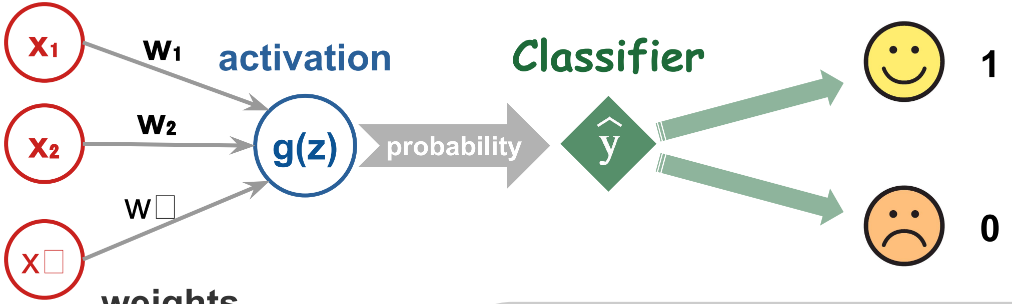


model performance is evaluated
on the unseen test set (generalization)

Logistic Regression

input data

Binary Classification

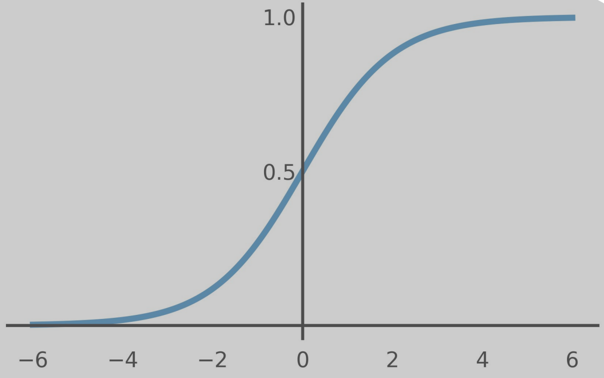


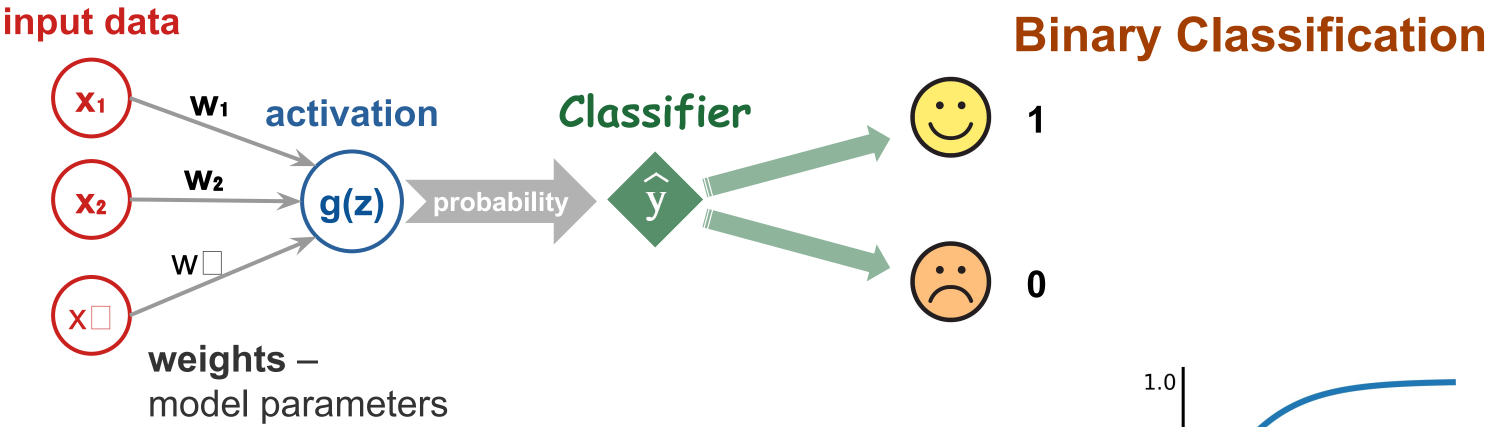
weights –
model parameters

$$w_1 x_1 + w_2 x_2 + \cdots = z$$

Sigmoid function

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

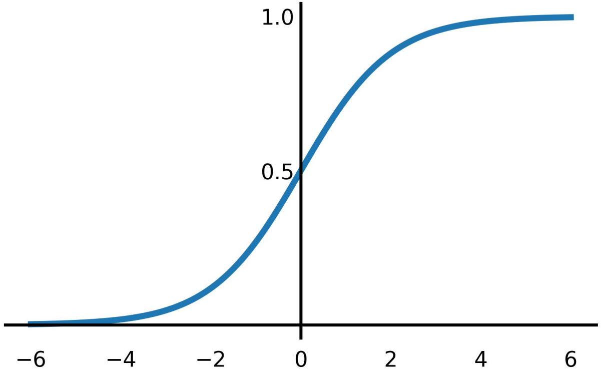




$$w_1 x_1 + w_2 x_2 + \cdots = z$$

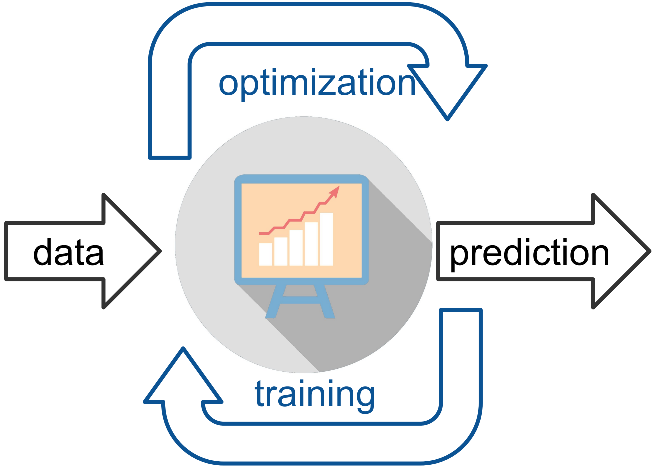
Sigmoid function

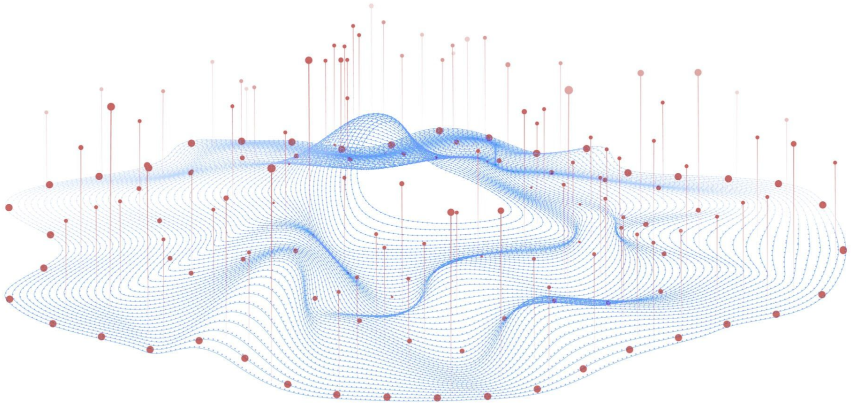
$$g(z) = \frac{1}{1 + e^{-z}}$$

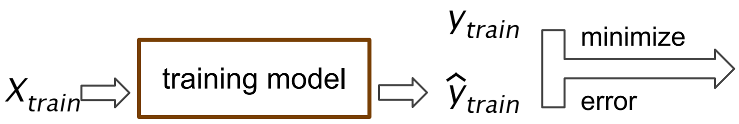


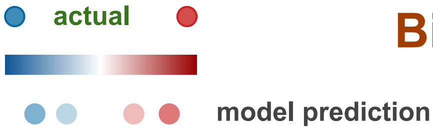
Optimization

Goal: minimizes the prediction error for binary classification









Binary Classification

prediction error

difference between what the model predicted and what the actual (true)

Binary Cross Entropy

M – number of instances

$\log$ – the natural logarithm

y – binary true label (0 or 1)

$\hat{y}$ – predicted probability

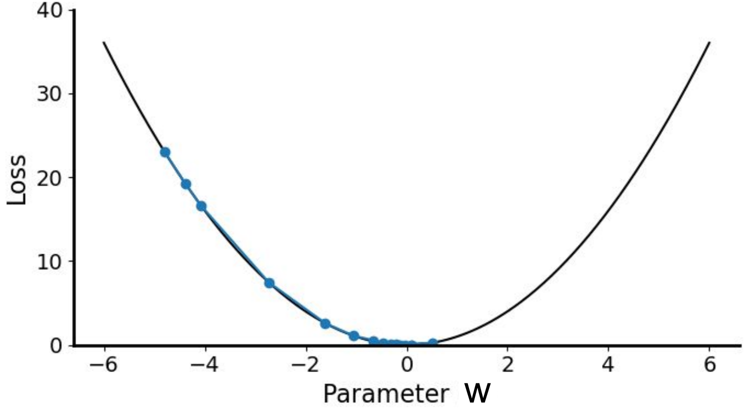
$$loss(y,\hat{y}) = -(y\log(\hat{y}) + (1-y)\log(1-\hat{y}))$$

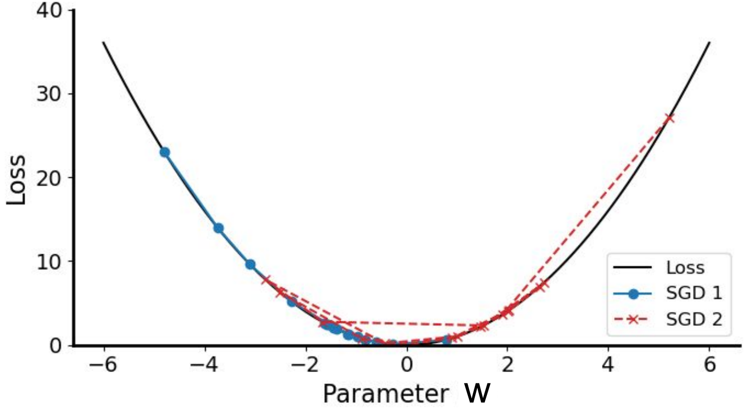
$$Loss(y,\hat{y}) = \frac{1}{M}\sum_{i=1}^{M} loss(y_i,\hat{y}_i)$$

Goal: minimize Loss function

Maths

Stochastic Gradient Descent





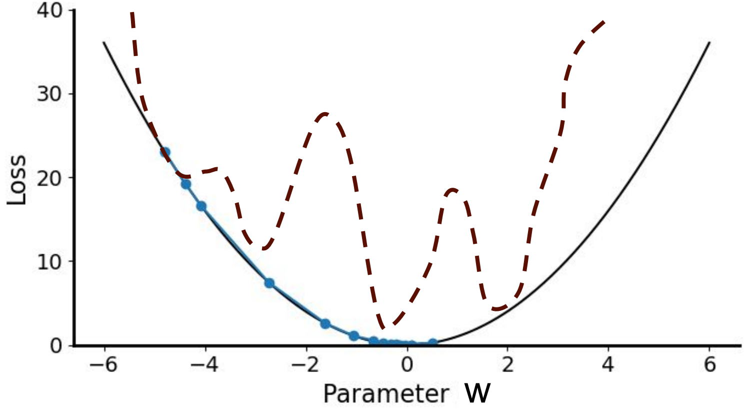
Stochastic Gradient Descent (SGD)

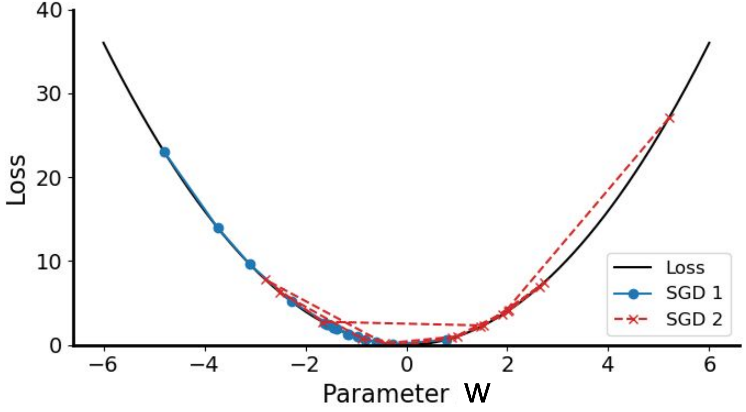
an iterative optimization method that approximates gradient descent by using a small randomly selected subset of data

$$w_{new} = w_{old} - \alpha \frac{\partial \mathrm{Loss}(y,\hat{y})}{\partial w}$$

α Learning rate

Maths





Stochastic Gradient Descent (SGD)

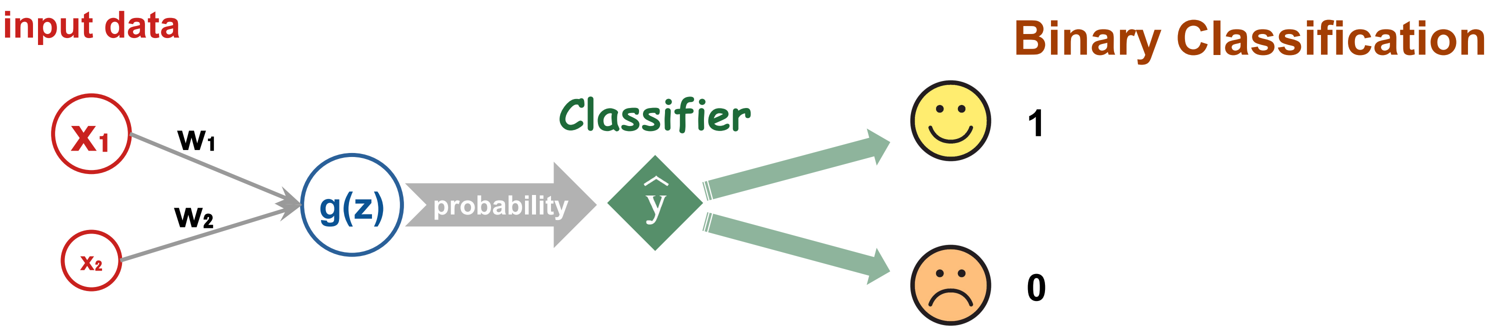
an iterative optimization method that approximates gradient descent by using a small randomly selected subset of data

$$w_{new} = w_{old} - \alpha \frac{\partial \text{Loss}(y, \hat{y})}{\partial w}$$

α Learning rate

Maths

Normalization



Rescaling numerical features to a common range (often [0, 1] or [-1, 1])

- Prevents features with large scales from dominating the model
- Improves convergence speed in optimization (e.g., gradient descent)

Min-Max Scaling: $$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Standardization: $$x_{scaled} = \frac{x - \mu}{\sigma}$$

μ mean

σ standard deviation

Evaluation

Performance Metrics

sensitivity

specificity

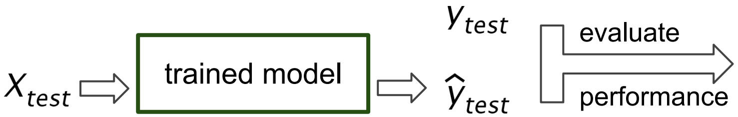
accuracy

AUROC

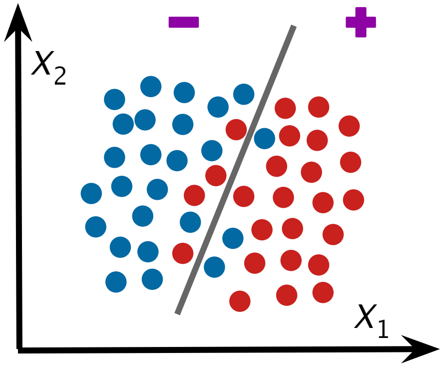
precision

F1–score

- ➔ Measures how well a trained model performs on **unseen test data**
- ➔ Estimates how the model **generalizes** beyond the **training data**



Classification: Evaluation Metrics



Confusion Matrix

		Model Prediction	
Ground Truth		**P**	**N**
	P	*TP*	*FN*
	N	*FP*	*TN*

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

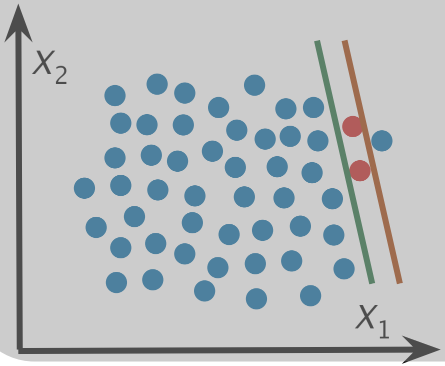
$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

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

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

21	4
3	22

Acc = 0.86
Sen = 0.84
Spe = 0.88

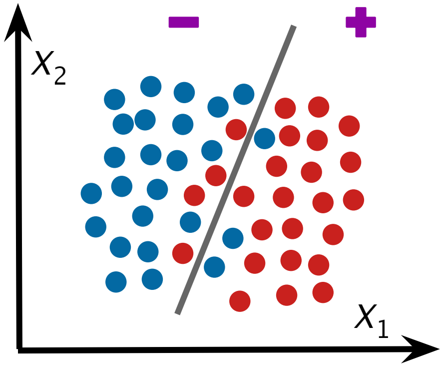


0	2
1	47

Acc = 0.940
Sen = 0.0
Spe = 0.979

2	0
1	47

Acc = 0.98
Sen = 1.0
Spe = 0.979



Confusion Matrix

		Model Prediction	
		P	N
Ground Truth	P	TP	FN
	N	FP	TN

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

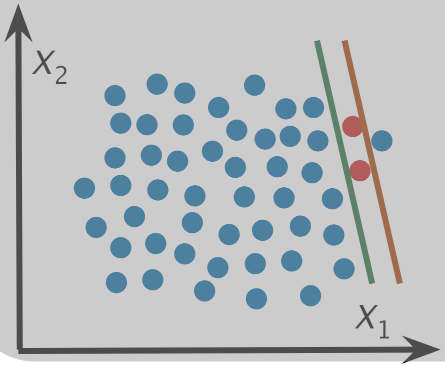
$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

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

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

21	4
3	22

Acc = 0.86
Sen = 0.84
Spe = 0.88



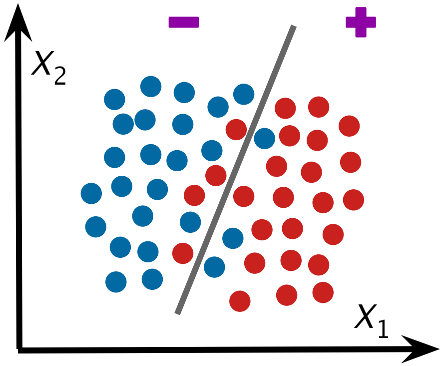
0	2
1	47

Acc = 0.940
Sen = 0.0
Spe = 0.979

2	0
1	47

Acc = 0.98
Sen = 1.0
Spe = 0.979

Classification: Evaluation Metrics



Confusion Matrix

Model Prediction

Ground Truth	P	N
P	*TP*	*FN*
N	*FP*	*TN*

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

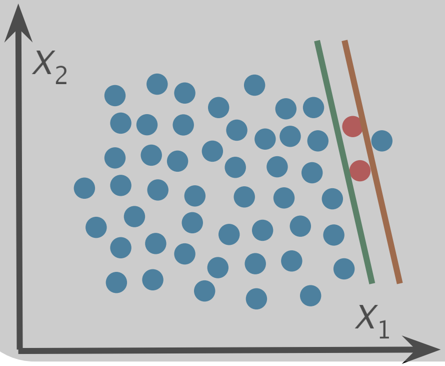
Accuracy $= \frac{TP+TN}{TP+TN+FP+FN}$

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

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

21	4
3	22

Acc = 0.86
Sen = 0.84
Spe = 0.88



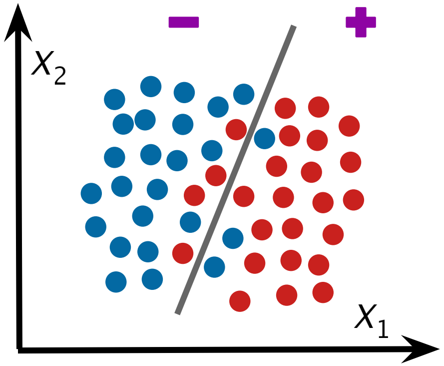
0	2
1	47

Acc = 0.940
Sen = 0.0
Spe = 0.979

2	0
1	47

Acc = 0.98
Sen = 1.0
Spe = 0.979

Classification: Evaluation Metrics



Confusion Matrix

		Model Prediction: P	Model Prediction: N
Ground Truth	P	*TP*	*FN*
	N	*FP*	*TN*

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

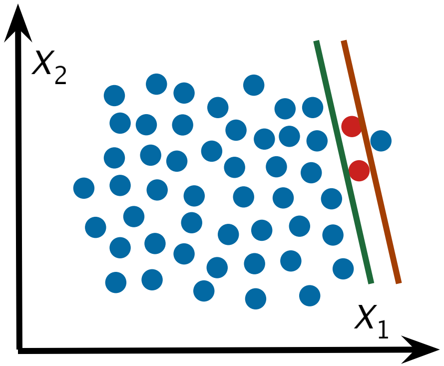
$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

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

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

21	4
3	22

Acc = 0.86
Sen = 0.84
Spe = 0.88



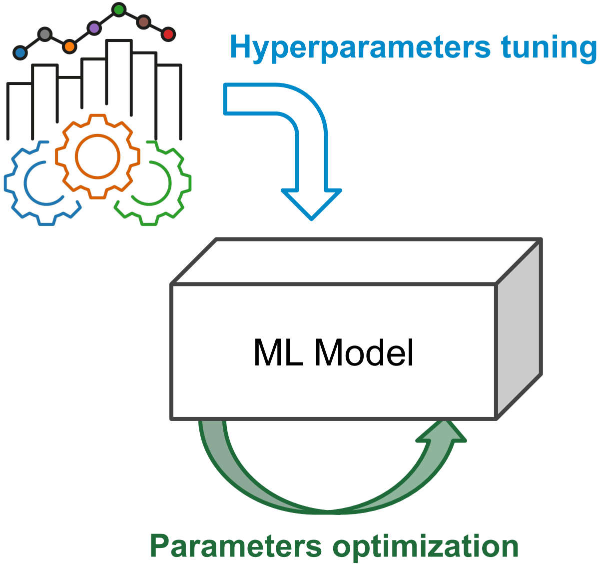
0	2
1	47

Acc = 0.940
Sen = 0.0
Spe = 0.979

2	0
1	47

Acc = 0.98
Sen = 1.0
Spe = 0.979

Hyperparameters



Hyperparameters vs. Parameters in ML

Hyperparameters

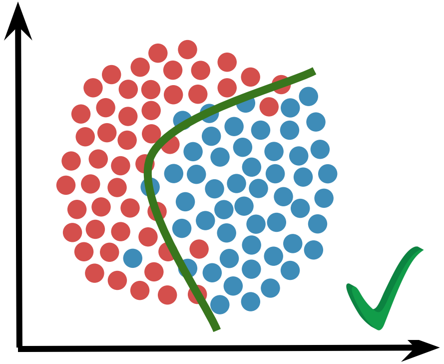
- ➢ Define the model's structure and training process
- ➢ Set before training (not learned from data)
- ➢ Examples: learning rate, regularization strength

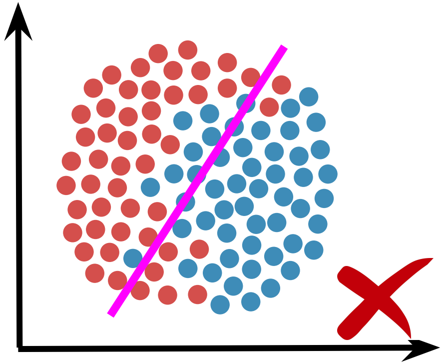
Parameters

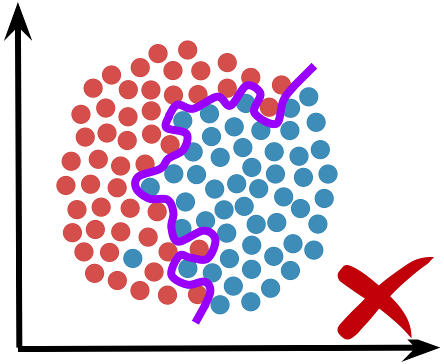
- ➢ Define how the model makes predictions
- ➢ Learned from data during training
- ➢ Examples: weights, biases in neural networks

Under- vs. Overfitting

Binary Classification







Good fit (robust model)

- Model captures patterns in data without excessive noise
- Performs well on both training and test dataset

Underfitting (simple model)

- Model is too simple, fails to capture patterns
- High error on training and test dataset

Overfitting (complex model)

- Model is too complex, memorizes training data
- Low training error but high test error

Bias-Variance Trade-off

Bias: Error from overly simplistic models (underfitting)

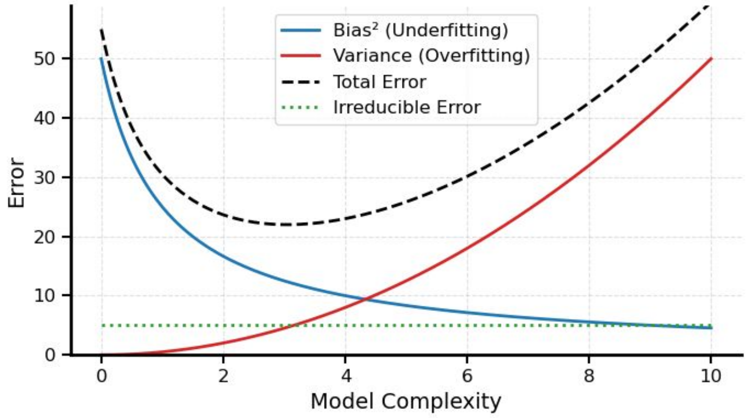
⟶ model misses important patterns

Variance: Error from overly complex models (overfitting)

⟶ model learns noise in the training data

Trade-off:

- ➢ Increasing model complexity: bias ↓ but variance ↑
- ➢ Goal = find the balance that minimizes **Total Error** on unseen data



$$\boldsymbol{Total\ Error} = Bias^2 + Variance + Irreducible\ Error$$

Regularization

- Prevents overfitting by penalizing large coefficients
- Adds a penalty term to the loss function
  - **L2 Regularization (Ridge)**

    Shrinks coefficients smoothly (reduces their magnitude)
  - **L1 Regularization (Lasso)**

    Introduces sparsity (feature selection) – some coefficients become 0
- Improves generalization and model robustness on unseen data

$$\mathrm{Loss}(y,\hat{y}) = \mathcal{L}$$

$$\mathcal{L}_{L2} = \mathcal{L} + \lambda \sum_{j=1}^{N} w_j^2$$

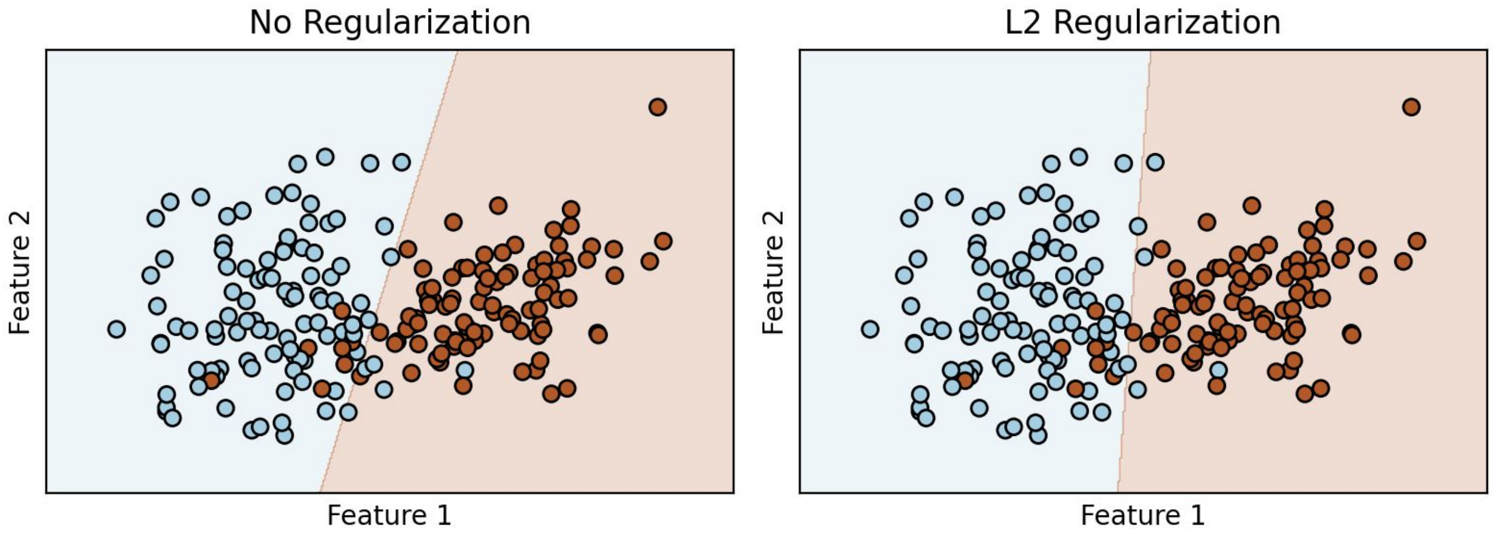
$$\mathcal{L}_{L1} = \mathcal{L} + \lambda \sum_{j=1}^{N} |w_j|$$

λ regularization strength (controls the penalty in the Loss function)

$\lambda = 0$ no regularization

Maths

Binary Classification



PIMA Diabetes Dataset

Goal: Predict whether a patient has diabetes (**No**/**Yes**)

Data: Medical measurements from PIMA Indian women (age ≥ 21)

- Features: age, BMI, blood pressure, glucose, insulin, pregnancies, etc.
- Target: Diabetes (0 = No, 1 = Yes)

Approach:

- **Logistic Regression** (binary classifier)
- Train/test split to evaluate generalization
- Metrics: **accuracy, sensitivity, specificity**

Patient Features → **Logistic Regression** → Probability P(Diabetes)

> **thr**: 1 – Yes

< **thr**: 0 – No

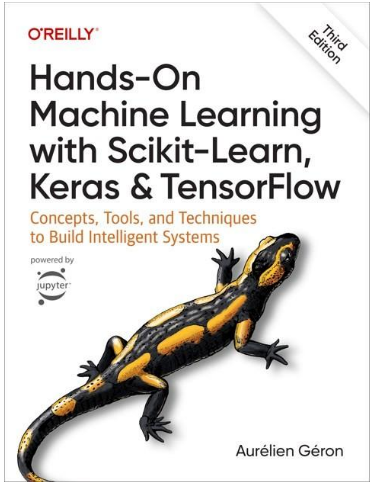
Insights:

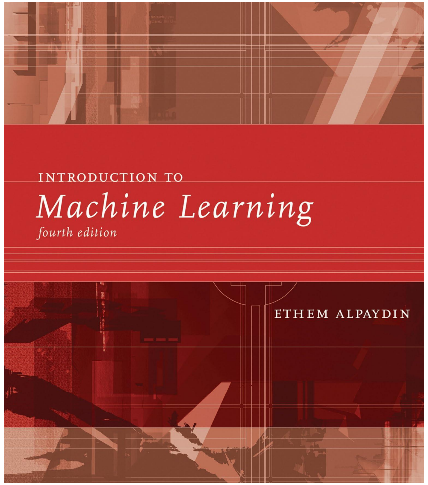
- Logistic regression provides **probability estimates** (risk of diabetes)
- Useful for **medical decision support** (screening, risk assessment)

Pima Indians Diabetes Database

Using the ADAP Learning Algorithm to Forecast the Onset of Diabetes Mellitus - PMC

Learning





- Machine Learning Specialization
- AI for Medicine Specialization
- Machine Learning Mastery
- StatQuest with Josh Starmer - YouTube

- Big Data and Machine Learning in Health Care | Artificial Intelligence | JAMA
- Disease Prediction by Machine Learning Over Big Data From Healthcare Communities | IEEE Journals & Magazine

THANK YOU!

Sina Sadeghi
Medical Data Science