LOVELY PROFESSIONAL UNIVERSITY

NAAC GRADE A++

INT395- SUPERVISED ML

Unit 2:Classification

Presented By: Blossom Kaler
Assistant Professor
SCSE, LPU

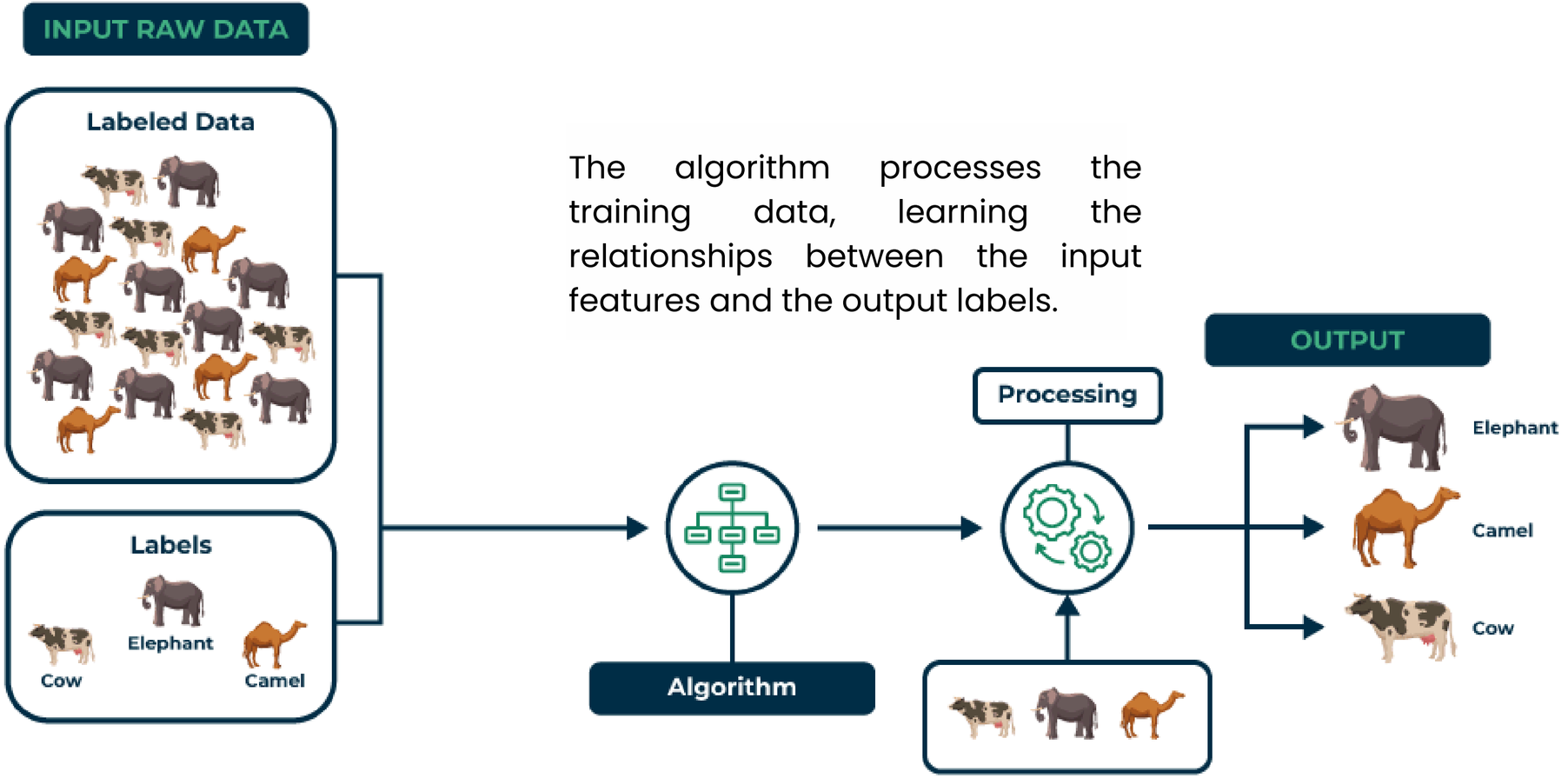
WHAT IS CLASSIFICATION?

- supervised learning task where the goal is to predict discrete labels (classes) for input data.
- Output variable is categorical (e.g., spam or not spam)
- Types: Binary and multiclass classification
- Requires labeled training data for learning patterns
- Used for image recognition, diagnosis, fraud detection, etc.

Spam Detection

CLASSIFICATION

Training data



The algorithm processes the training data, learning the relationships between the input features and the output labels.

CONFUSION MATRIX

- A table used to evaluate classification model performance.
- Compares actual vs predicted classes.
- Components:
  TP (True Positive): Correctly predicted positive
  TN (True Negative): Correctly predicted negative
  FP (False Positive): Incorrectly predicted positive
  FN (False Negative): Incorrectly predicted negative

- Helps identify types of classification errors.
- Basis for calculating precision, recall, accuracy, etc.

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

EVALUATION METRICS

- Used to measure model performance
- Essential for comparing models
- Different tasks → Different metrics
- Accuracy alone is not always sufficient

Helps understand:

- How well the model predicts
- Types of errors
- Strengths & weaknesses
- Guides model improvement & selection
- Depends on business needs (FP vs FN cost)

Assessments	Formula
Precision (P)	$\frac{\text{TP}}{\text{TP+FP}}$
Recall (R)	$\frac{\text{TP}}{\text{TP+FN}}$
F1-score	$2 \times \frac{P \times R}{P+R}$
Accuracy	$\frac{\text{TP+TN}}{\text{TP+TN+FP+FN}}$

ACCURACY

- Proportion of correct predictions.
- Formula: (TP + TN) / (TP + TN + FP + FN)
- Easy to understand and widely used.
- Works well when classes are balanced.

- Image Classification – Cats vs Dogs
- If dataset has similar numbers of cat and dog images, accuracy gives a reliable measure of how well the model distinguishes them.

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

PRECISION

- Measures correctness of Positive predictions
- **“Out of all positive predictions, how many are actually positive?”**
- High Precision → few false alarms

- Spam Detection
- Wrongly marking important emails as spam (FP) is very costly → needs high precision

- When Preferred? - When False Positive cost is high
- Spam detection
- Job candidate filtering
- Email phishing detection

$$Precision = \frac{True\ Positive(TP)}{True\ Positive(TP) + False\ Positive(FP)}$$

RECALL

- Measures how well model finds actual positives
- **“Of all actual positives, how many were captured?”**
- High Recall → few missed positives

- Cancer Detection
- Missing real patients (FN) is risky → require high recall

- When Preferred? - When False Negative cost is high
- Disease diagnosis
- Security intrusion detection
- Missing fraud transactions

$$\text{Recall} = \frac{\text{True Positive (TP)}}{\text{True Positive (TP)} + \text{False Negative (FN)}}$$

F1-SCORE

- Harmonic mean of Precision & Recall
- Good combined measure
- Best for imbalanced datasets

- Fraud detection
- Both missed fraud (FN) and wrongly blocked transactions (FP) are harmful
- F1 gives balanced evaluation

- When Preferred? - When you need a balance between Precision & Recall
- When classes are imbalanced

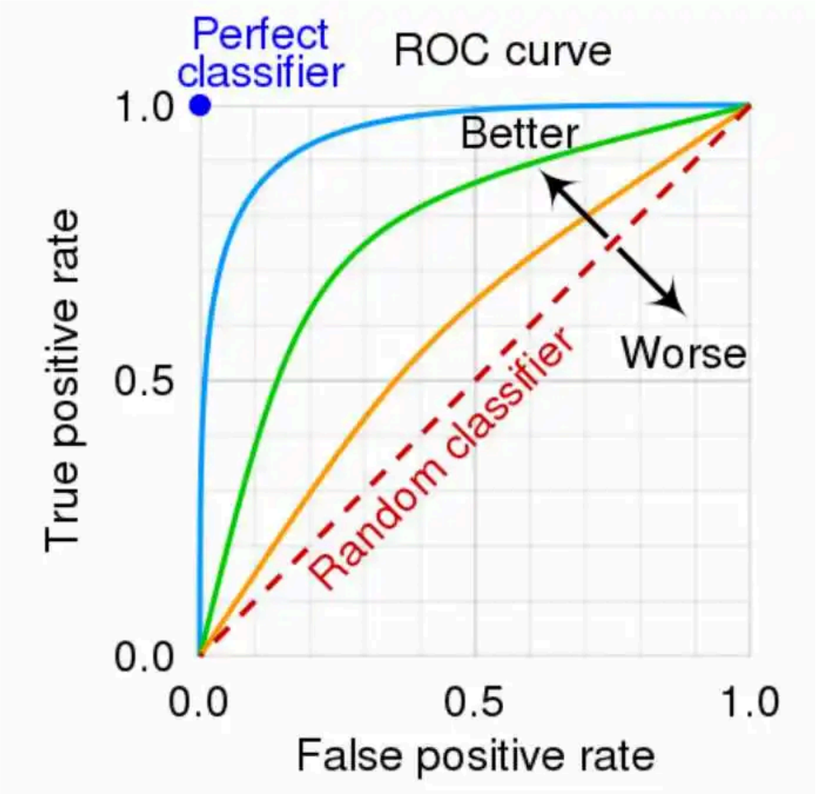
$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

ROC-AUC CURVE

- Receiver Operating Characteristic curve
- ROC curve shows trade-off between sensitivity and specificity.
- Plots True Positive Rate (TPR) vs. False Positive Rate (FPR).

Intuition:

- TPR = "Of all real positives, how many did we catch?"
- FPR = "Of all real negatives, how many did we wrongly mark as positive?"



$$\text{True Positive Rate (TPR)} = \frac{TP}{TP + FN}$$

also called sensitivity/recall/hit rate

$$\text{False Positive Rate (FPR)} = \frac{FP}{FP + TN}$$

also called fall out

TPR V/S FPR

Cases where High TPR is important

- Medical diagnosis (e.g., cancer, COVID tests)
- Missing a sick patient (false negative) is dangerous.
- Even if some healthy people are flagged wrongly (false positives), doctors can double-check.

Cases where Low FPR is important

- Criminal justice (face recognition for suspects)
- A high FPR means many innocent people are flagged as criminals → unacceptable.

$$\text{True Positive Rate (TPR)} = \frac{TP}{TP + FN}$$

also called sensitivity/recall/hit rate

$$\text{False Positive Rate (FPR)} = \frac{FP}{FP + TN}$$

also called fall out

PERCEPTRON

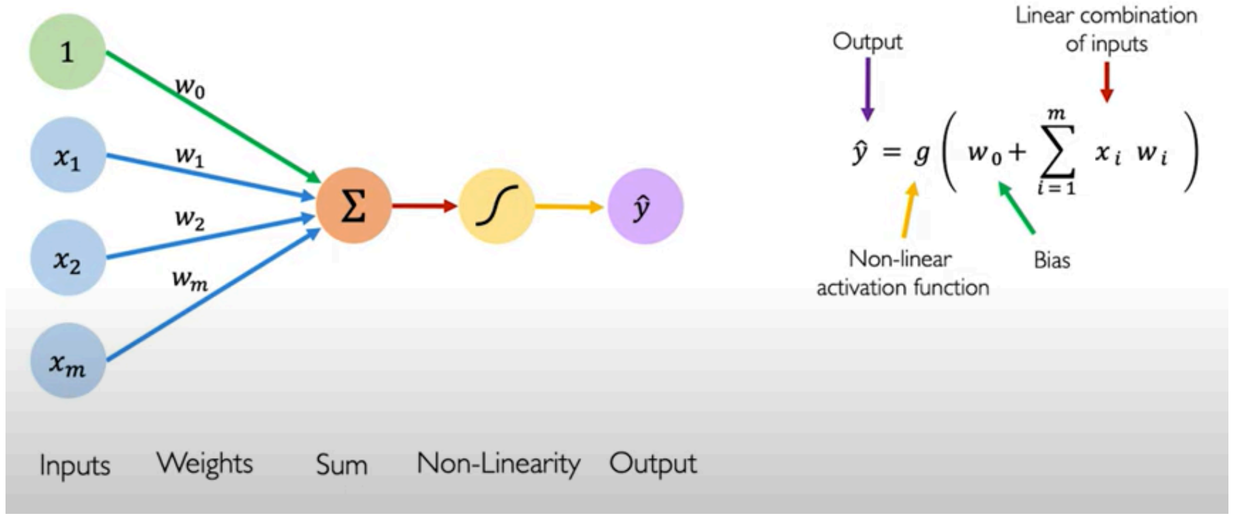
- Simplest form of a neural network
- Linear classifier
- Takes multiple inputs, computes weighted sum, applies a step function
- Works only for linearly separable data
- No probability output
- Sensitive to outliers

Equation:

$$y = f(w_1x_1 + w_2x_2 + ... + w_nx_n + b)$$

where

$$f(z) = \begin{cases} 1, & z > 0 \\ 0, & z \leq 0 \end{cases}$$



LOGISTIC REGRESSION

- A supervised classification algorithm
- Predicts probability of class = 1
- Uses sigmoid function to map linear combination → probability
    - Linear part:

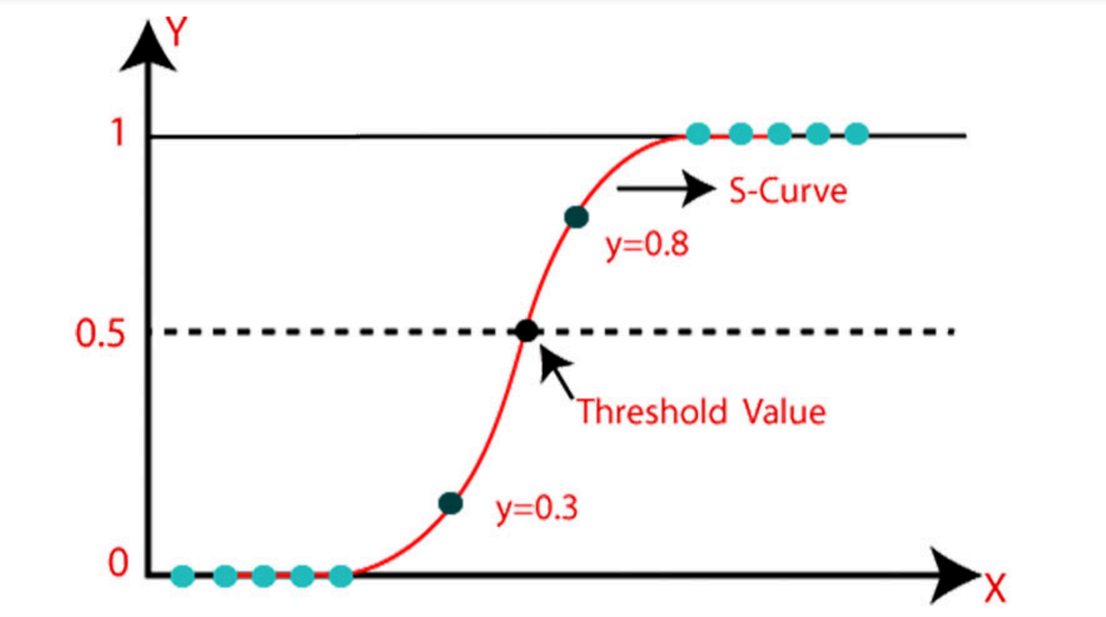
$$z = w_1x_1 + w_2x_2 + \cdots + b$$

    - Logistic output:

$$h_\theta(x) = \sigma(z)$$

Where:

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



LOGISTIC REGRESSION FOR MULTICLASS CLASSIFICATION

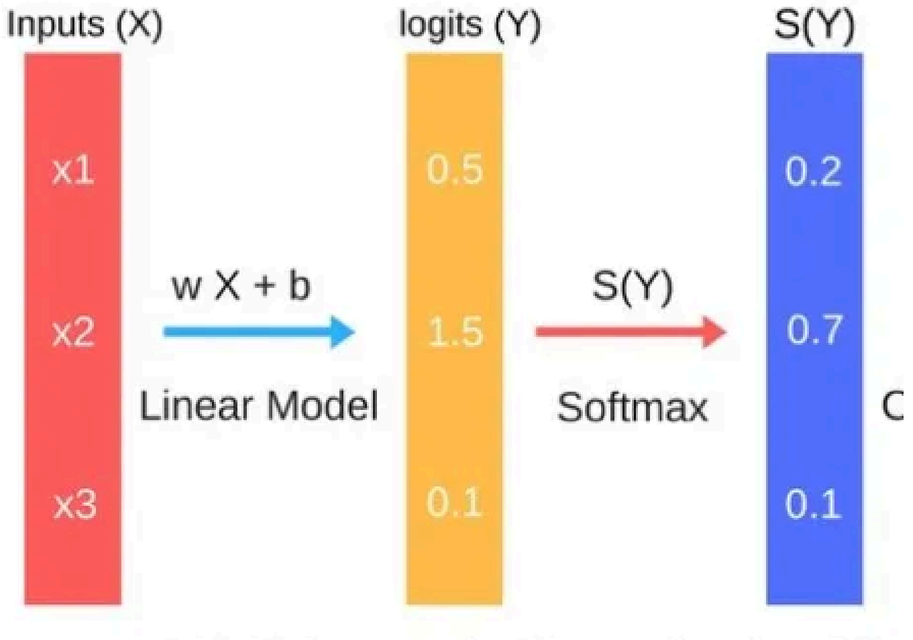
- Extension of logistic regression for 3+ classes
- Predicts probability for each class

Uses **Softmax function**:

$$P(y = i|x) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

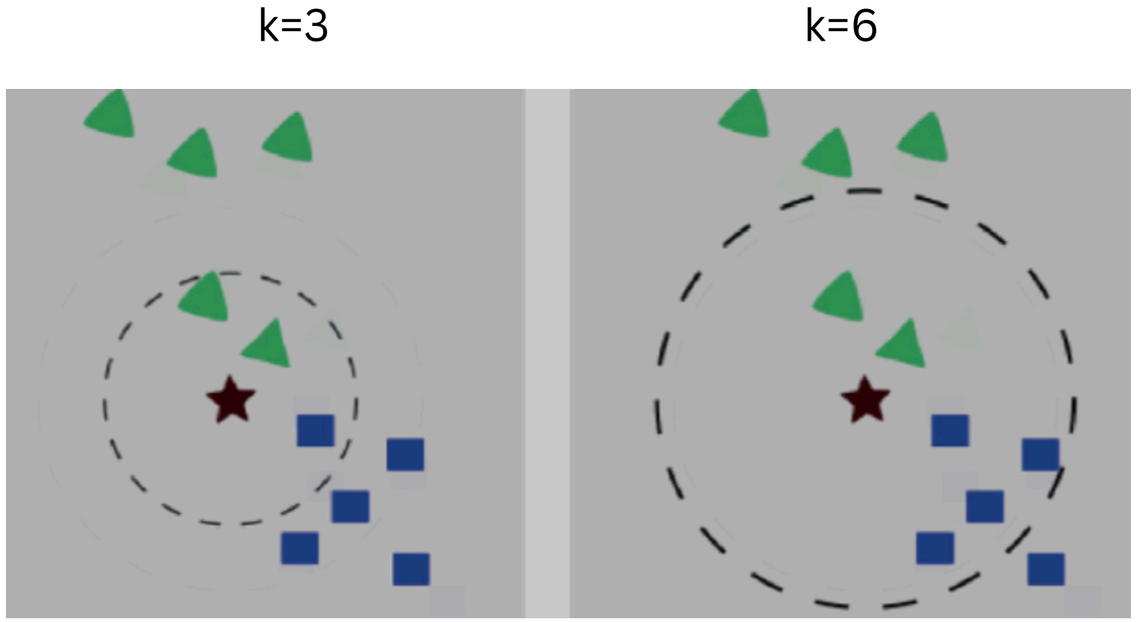
Decision rule:

$$\arg\max_i (P(y = i|x))$$



K-NEAREST NEIGHBORS

- classifies based on majority vote among k closest data points.
- No training phase
- Distance metric (like Euclidean) is key
- Sensitive to irrelevant features and scaling



KNN NUMERICAL

Given data

Brightness	Saturation	Class
40	20	Red
50	50	Blue
60	90	Blue
10	25	Red
70	70	Blue
60	10	Red
25	80	Blue

New Data Point, K=5

Brightness	Saturation	Class
20	35	?

Here's the formula: $\sqrt{(X_2-X_1)^2+(Y_2-Y_1)^2}$

Where:

- X_2 = New entry's brightness (20).
- X_1= Existing entry's brightness.
- Y_2 = New entry's saturation (35).
- Y_1 = Existing entry's saturation.

KNN NUMERICAL

Distance #1

For the first row, d1:

Brightness	Saturation	Class
40	20	Red

$d1 = \sqrt{(20 - 40)^2 + (35 - 20)^2}$

$= \sqrt{400 + 225}$

$= \sqrt{625}$

$= 25$

Brightness	Saturation	Class	Distance
40	20	Red	25
50	50	Blue	33.54
60	90	Blue	68.01
10	25	Red	10
70	70	Blue	61.03
60	10	Red	47.17
25	80	Blue	45

KNN NUMERICAL

5 nearest Neighbors

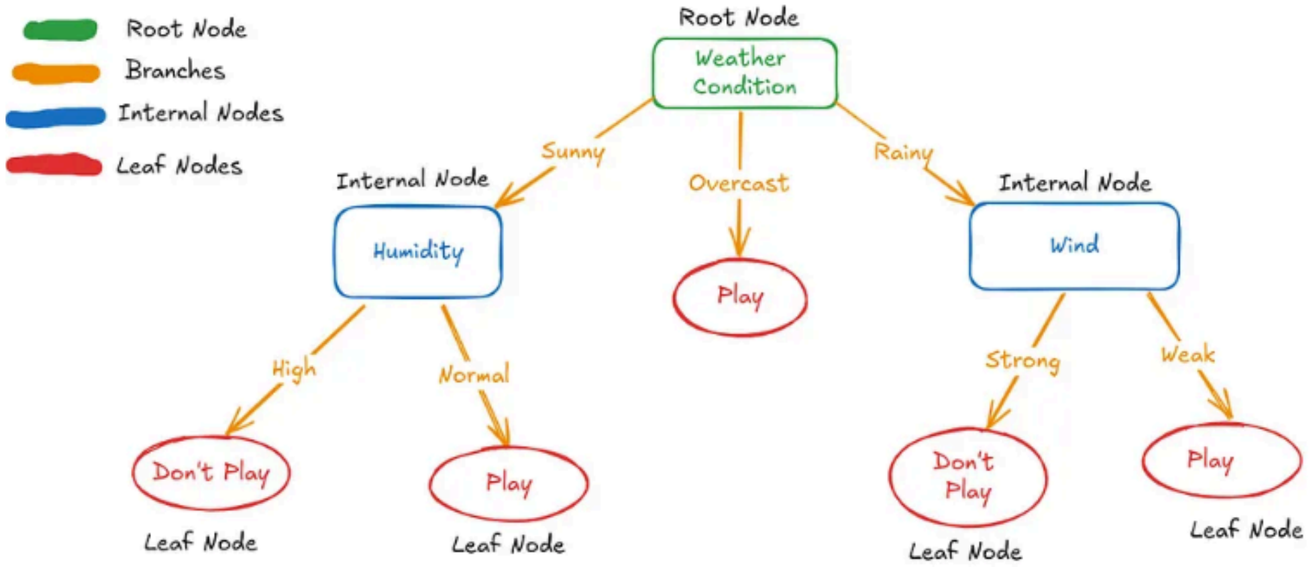
Brightness	Saturation	Class	Distance
10	25	Red	10
40	20	Red	25
50	50	Blue	33.54
25	80	Blue	45
60	10	Red	47.17

- the majority class within the 5 nearest neighbors to the **new entry is Red.**
- Therefore, we'll classify the new entry as Red.

Brightness	Saturation	Class
20	35	Red

DECISION TREE

- it is a flowchart-like structure used for classification by splitting data based on feature values.
- helps us to make decisions by mapping out different choices and their possible outcomes.



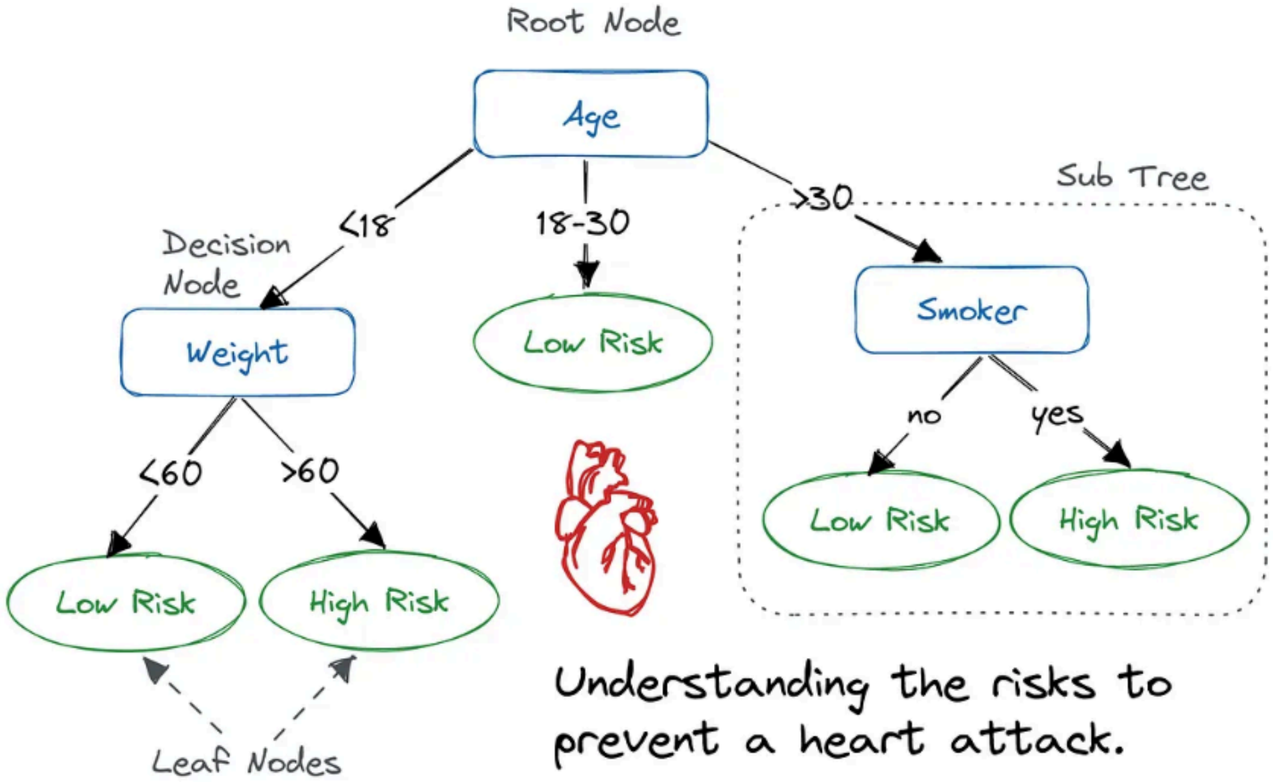
PlayTennis: training examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

DECISION TREE

Algorithms there to build a decision tree:

- **ID3** (Iterative Dichotomiser 3) — This uses entropy and information gain as metric.
- **CART** (Classification and Regression Trees) — This uses Gini impurity as the metric.



DECISION TREE NUMERICAL - ID3

$$Entropy(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

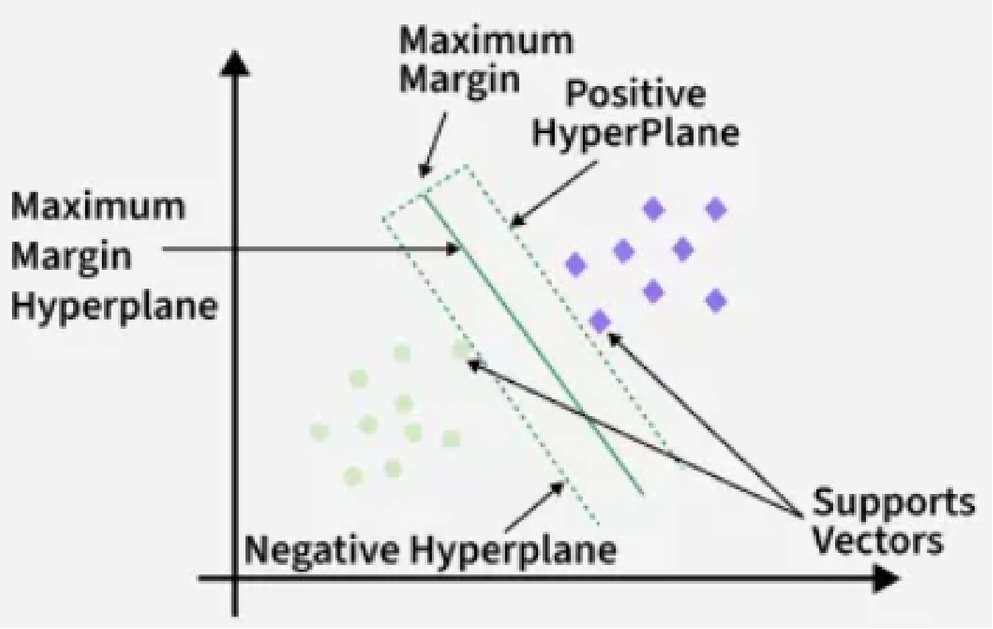
$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Refer decision tree numerical shared with this ppt!

SUPPORT VECTOR MACHINE

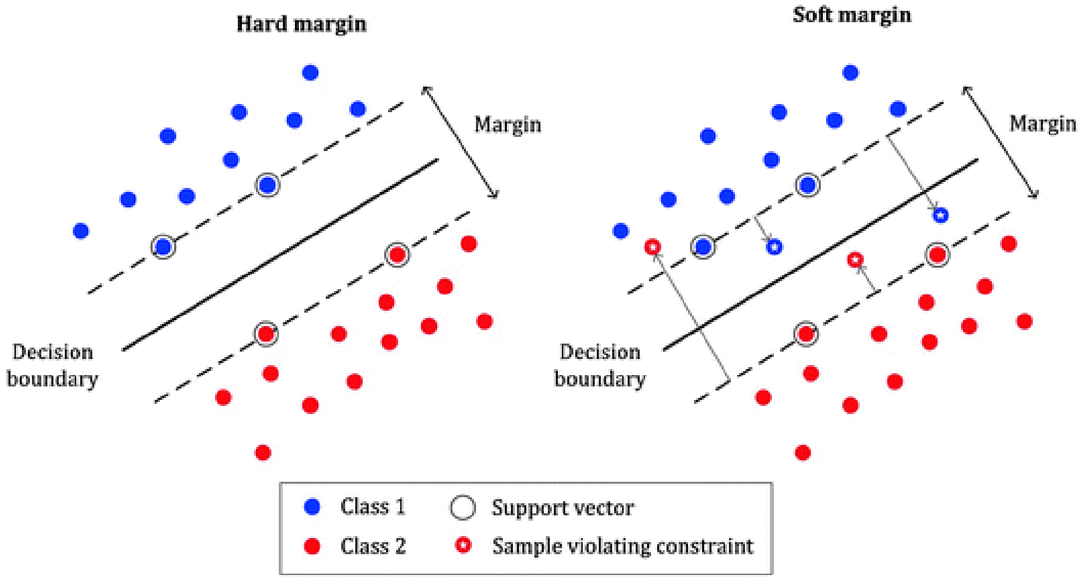
- Find a line/plane that best separates data into classes
- Choose the boundary with the largest margin
- More margin → better generalization
- Uses only “important” data points: support vectors

- Hyperplane → Decision boundary
- Margin → Distance between hyperplane & closest points
- Support Vectors → Points that lie closest to margin; influence boundary
- Goal: Maximize margin



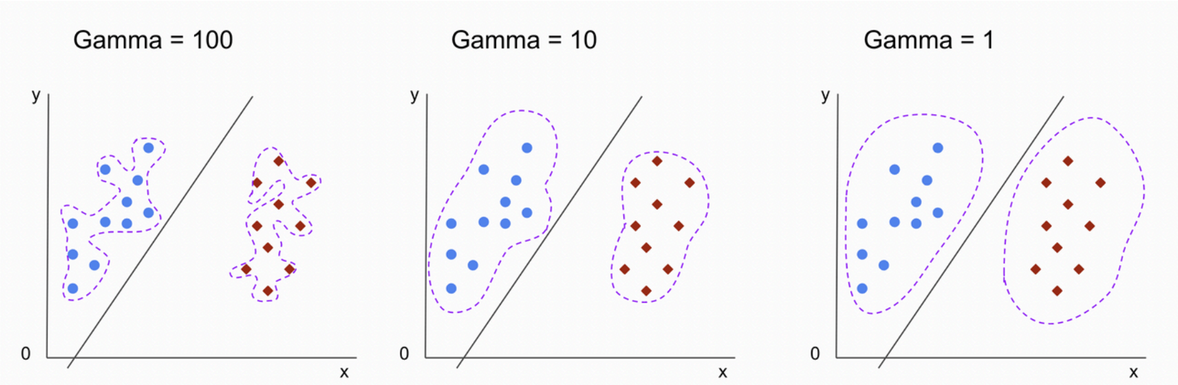
SOFT MARGIN VS HARD MARGIN

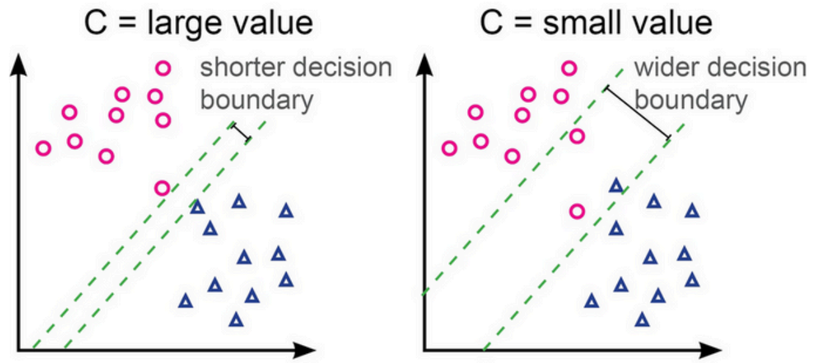
- **Hard Margin:**
- No misclassification allowed
- Works only if data is perfectly separable
- **Soft Margin:**
- Allows misclassification
- Better real-world performance
- Controlled by hyperparameter C
- High C → strict margin, fewer mistakes → may overfit
- Low C → flexible margin, more mistakes → may underfit



GAMMA V/S C PARAMETER

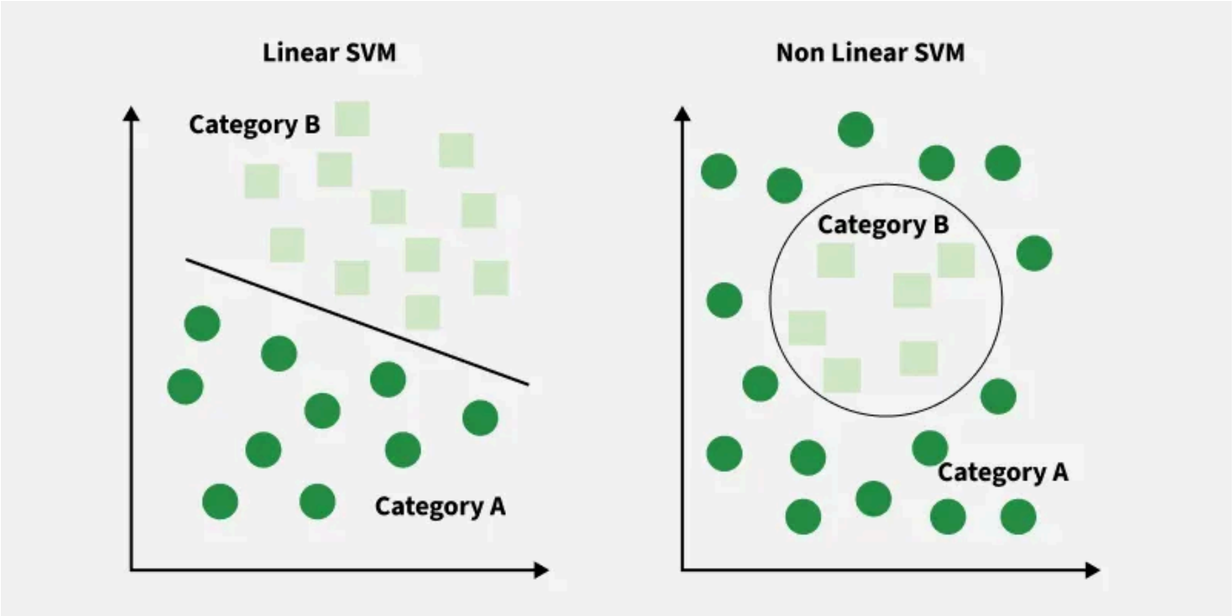
- **C controls margin strictness:**
- High C → very strict, tries to classify every point correctly, small margin, risk of overfitting.
- Low C → more relaxed, allows some misclassification, larger margin, better generalization.
- **Gamma controls boundary complexity:**
- High gamma → each point has a small influence area, boundary becomes highly curved and complex (overfitting).
- Low gamma → broader influence, smoother decision boundary (underfitting).
- **Key Difference:**
- C adjusts how wide the margin should be.
- Gamma adjusts how complex the decision boundary should be.





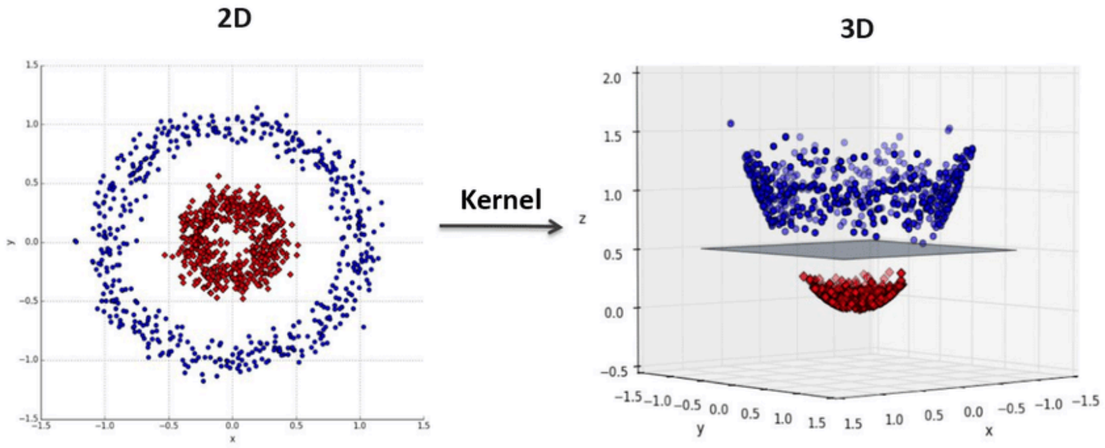
LINEAR V/S NON-LINEAR SVM

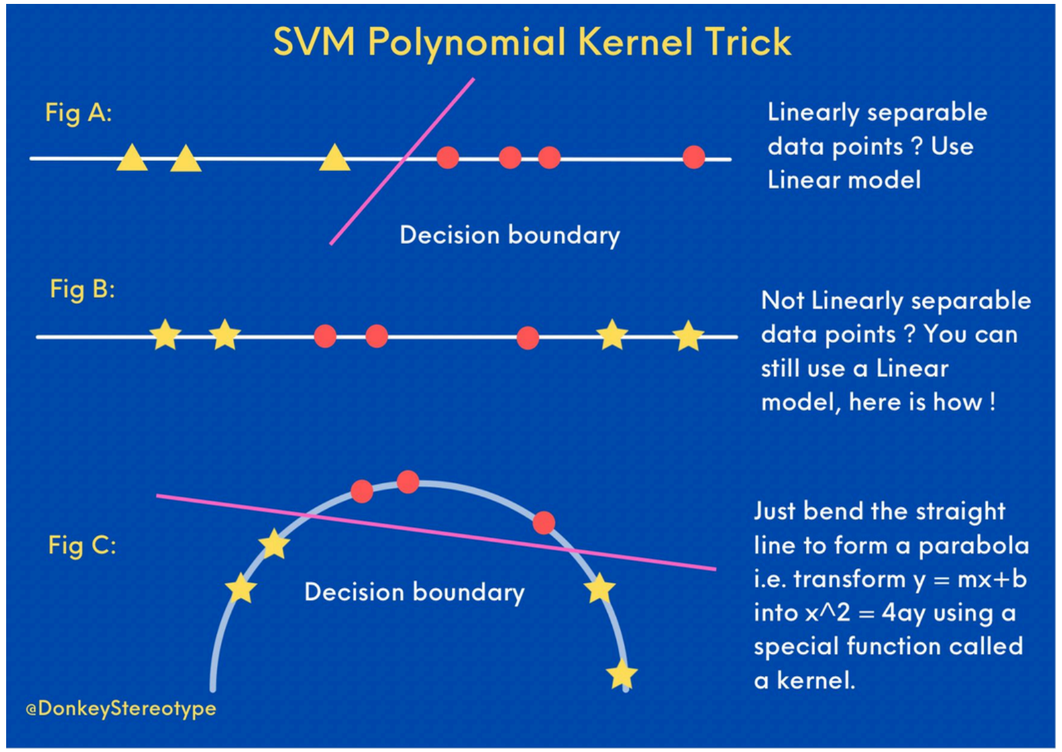
- Assumes data is linearly separable
- Finds optimal separating hyperplane
- Max margin leads to better performance
- Simple & interpretable



KERNEL TRICK

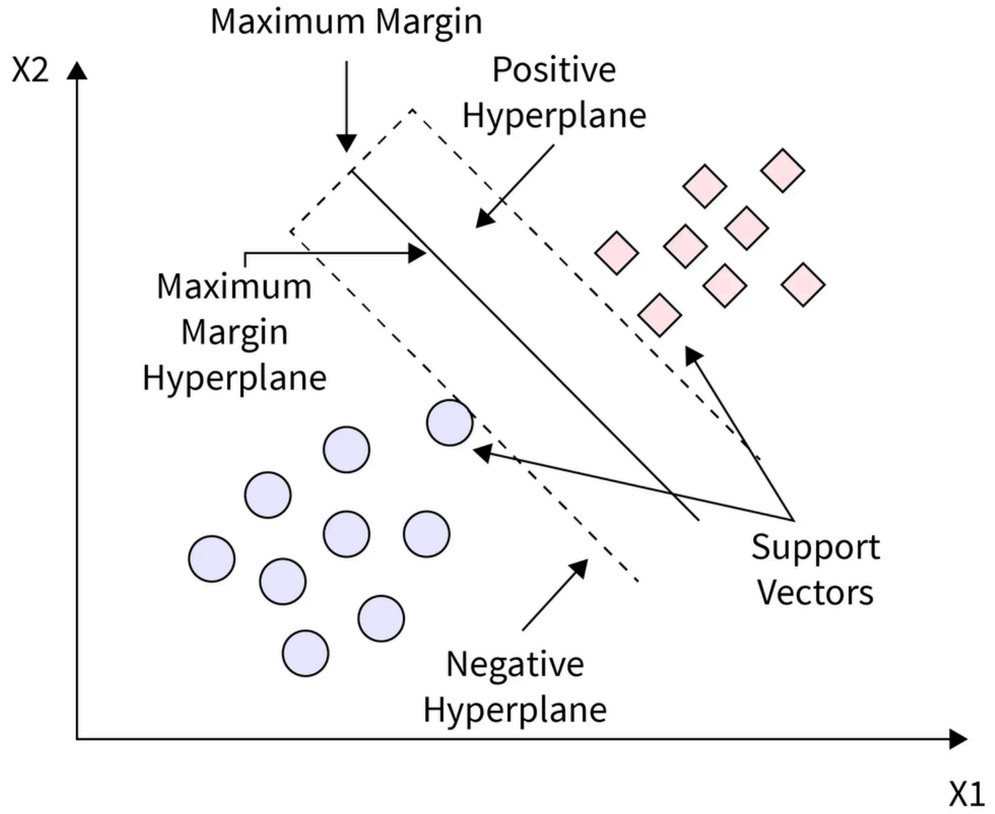
- Kernel = a function that transforms data into a higher-dimensional space.
- Allows SVM to find linear separation in transformed space.
- Useful when data is not linearly separable.
- “kernel trick” → avoid explicitly creating new features.
- Different kernels capture different types of patterns.





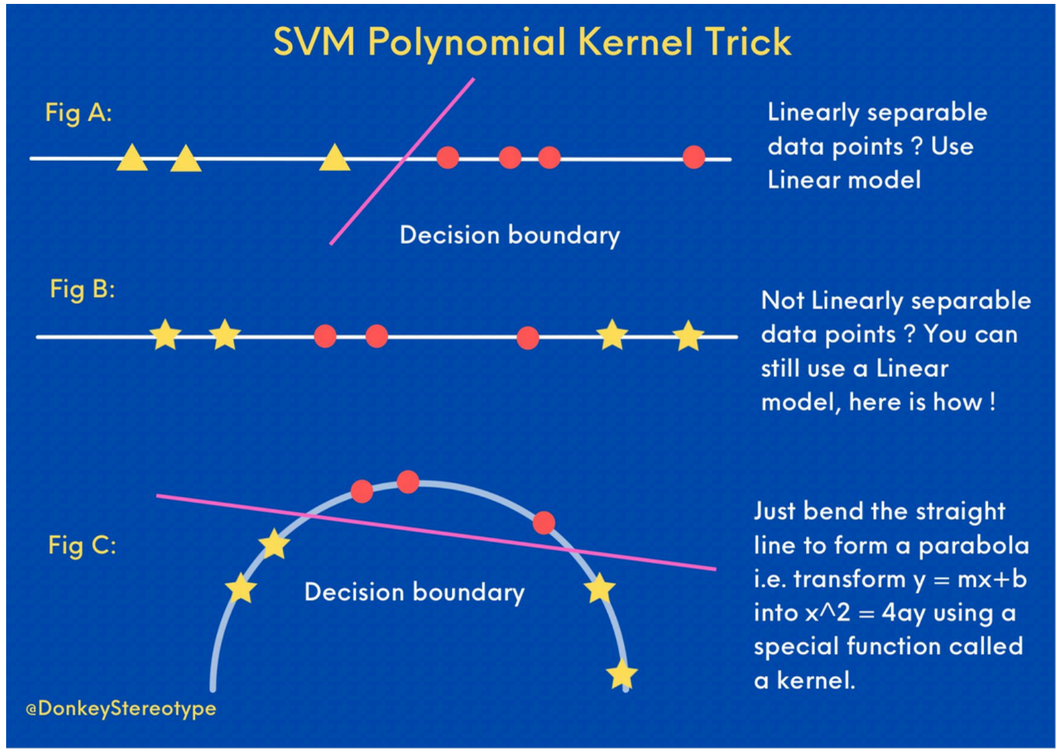
LINEAR KERNEL

- Simplest SVM kernel; uses original features without transformation.
- Creates a straight-line (flat) decision boundary.
- Works best when data is already linearly separable or nearly separable.
- Very fast, easy to interpret, and less prone to overfitting.
- Ideal for high-dimensional datasets like text classification.



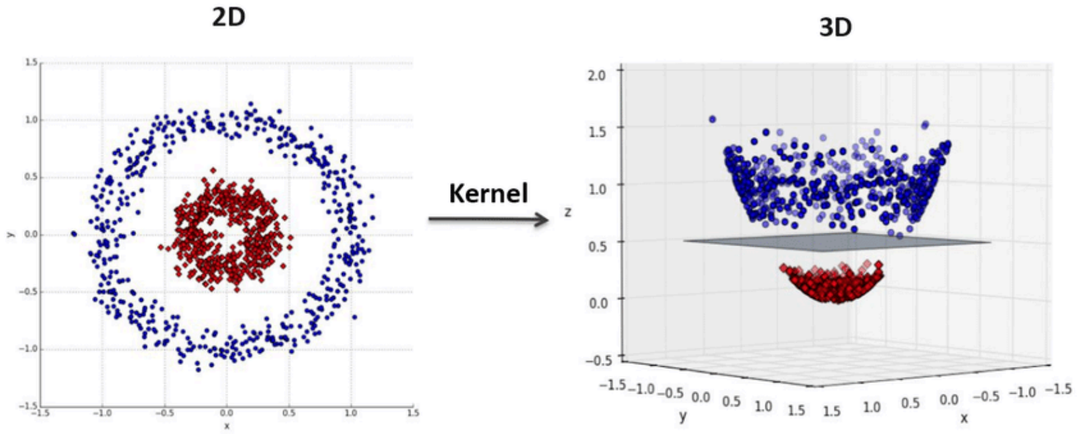
POLYNOMIAL KERNEL

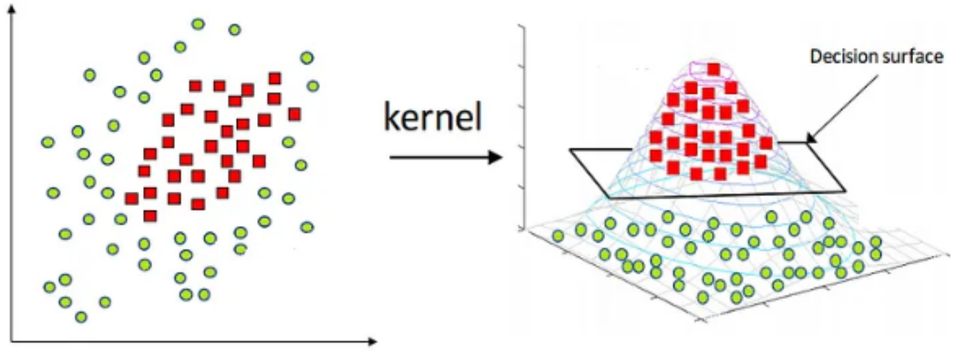
- Creates feature interactions (e.g., combinations of features).
- Helps capture moderately complex, curved decision boundaries.
- Useful when relationships between features are non-linear but still structured.
- More flexible than linear but less flexible than RBF.



RBF (GAUSSIAN) KERNEL

- The most commonly used kernel in practice.
- Creates a highly flexible decision boundary that can adapt to complex patterns.
- Excellent when classes are intertwined or irregularly shaped.
- Can handle almost any non-linear structure.
- Needs careful tuning to avoid overfitting.





NAIVE BAYES

- probabilistic classifier based on Bayes' Theorem
- It is named as "Naive" because it assumes the presence of one feature does not affect other features.
- All features are assumed to contribute equally to the prediction of the class label.

$$P(y|X) = \frac{P(X|y) \cdot P(y)}{P(X)}$$

- $P(y|X)$: Posterior probability, probability of class y given features X
- $P(X|y)$: Likelihood, probability of features X given class y
- $P(y)$: Prior probability of class y
- $P(X)$: Marginal likelihood or evidence

NAIVE BAYES WORKING

- $X=(x1,x2,...,xn)$ is the feature vector.
- Since, we assume that all features are independent given the class, therefore:

$$P(x_1, x_2, ..., x_n|y) = P(x_1|y) \cdot P(x_2|y) \cdots P(x_n|y)$$

Thus, Bayes' theorem becomes:

$$P(y|x_1, ..., x_n) = \frac{P(y) \cdot \prod_{i=1}^{n} P(x_i|y)}{P(x_1)P(x_2)...P(x_n)}$$

Since the denominator is constant for a given input, we can write:

$$P(y|x_1, ..., x_n) \propto P(y) \cdot \prod_{i=1}^{n} P(x_i|y)$$

We compute the posterior for each class y and choose the class with the highest probability:

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

NAIVE BAYES NUMERICAL EXAMPLE

PlayTennis: training examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

$$P(PlayTennis = yes) = 9/14 = .64$$

$$P(PlayTennis = no) = 5/14 = .36$$

Outlook	Y	N
sunny	2/9	3/5
overcast	4/9	0
rain	3/9	2/5
Tempreature		
hot	2/9	2/5
mild	4/9	2/5
cool	3/9	1/5

Humidity	Y	N
high	3/9	4/5
normal	6/9	1/5
Windy		
Strong	3/9	3/5
Weak	6/9	2/5

New Input:

$\langle Outlook = sunny, Temperature = cool, Humidity = high, Wind = strong \rangle$

NAIVE BAYES NUMERICAL EXAMPLE

New Input:

$$\langle Outlook = sunny, Temperature = cool, Humidity = high, Wind = strong \rangle$$

Outlook	Y	N
sunny	2/9	3/5
overcast	4/9	0
rain	3/9	2/5
Tempreature		
hot	2/9	2/5
mild	4/9	2/5
cool	3/9	1/5

Humidity	Y	N
high	3/9	4/5
normal	6/9	1/5
Windy		
Strong	3/9	3/5
Weak	6/9	2/5

$$P(PlayTennis = yes) = 9/14 = .64$$

$$P(PlayTennis = no) = 5/14 = .36$$

$$v_{NB}(yes) = P(yes)\ P(sunny|yes)\ P(cool|yes)\ P(high|yes)\ P(strong|yes) = .0053$$

$$v_{NB}(no) = P(no)\ P(sunny|no)\ P(cool|no)\ P(high|no)\ P(strong|no) = .0206$$

Normalized Probabilities:

$$v_{NB}(yes) = \frac{v_{NB}(yes)}{v_{NB}(yes)+v_{NB}(no)} = 0.205 \qquad v_{NB}(no) = \frac{v_{NB}(no)}{v_{NB}(yes)+v_{NB}(no)} = 0.795$$

NAIVE BAYES - DIY

Example No.	Color	Type	Origin	Stolen?
1	Red	Sports	Domestic	Yes
2	Red	Sports	Domestic	No
3	Red	Sports	Domestic	Yes
4	Yellow	Sports	Domestic	No
5	Yellow	Sports	Imported	Yes
6	Yellow	SUV	Imported	No
7	Yellow	SUV	Imported	Yes
8	Yellow	SUV	Domestic	No
9	Red	SUV	Imported	No
10	Red	Sports	Imported	Yes

New Input:

Color	Type	Origin	Stolen
Red	SUV	Domestic	?

NAIVE BAYES -DIY

		Stolen?	
		P(Yes)	P(No)
Color	Red	3/5	2/5
	Yellow	2/5	3/5

		Stolen?	
		P(Yes)	P(No)
Type	Sports	4/5	2/5
	SUV	1/5	3/5

		Stolen?	
		P(Yes)	P(No)
Origin	Domestic	2/5	3/5
	Imported	3/5	2/5

$$P(Yes \mid X) = P(Red \mid Yes) * P(SUV \mid Yes) * P(Domestic \mid Yes) * P(Yes)$$

$$P(No \mid X) = P(Red \mid No) * P(SUV \mid No) * P(Domestic \mid No) * P(No)$$

THANK YOU