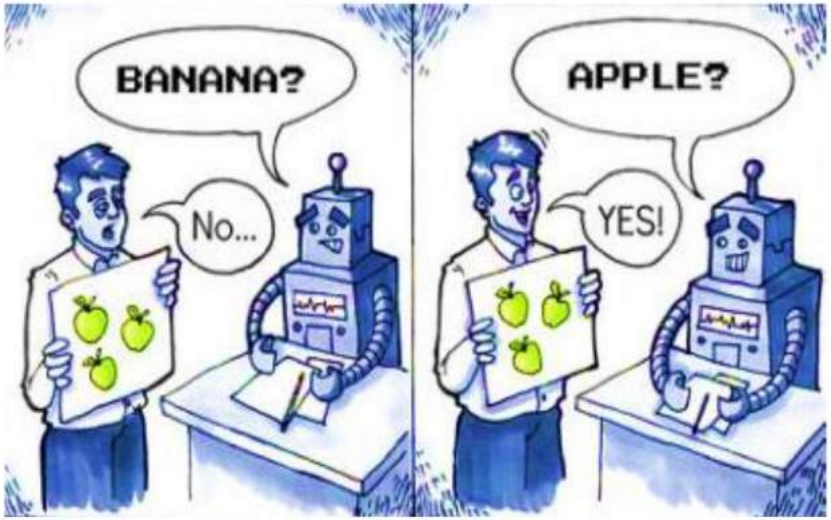
Supervised Learning

Supervised Learning

- Supervised Learning is the process of making an algorithm to learn to map an input to a particular output.
- Supervised Learning is possible with correctly labelled dataset.
- Discover patterns in the data that relate data attributes with a target (class) attribute.
- Train your data set to such extent that it becomes easy to find out the result of unknown inputs by making prediction of the class.



Example

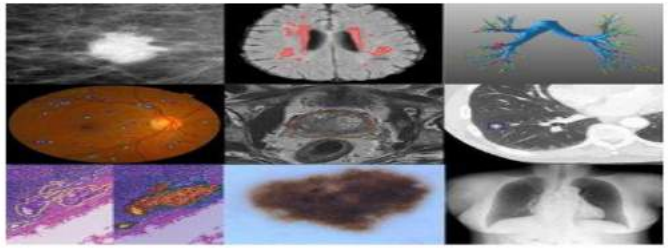
Relationship

"Dog"

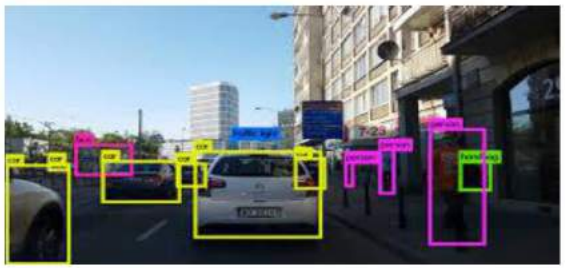
Relationship

"Cat"

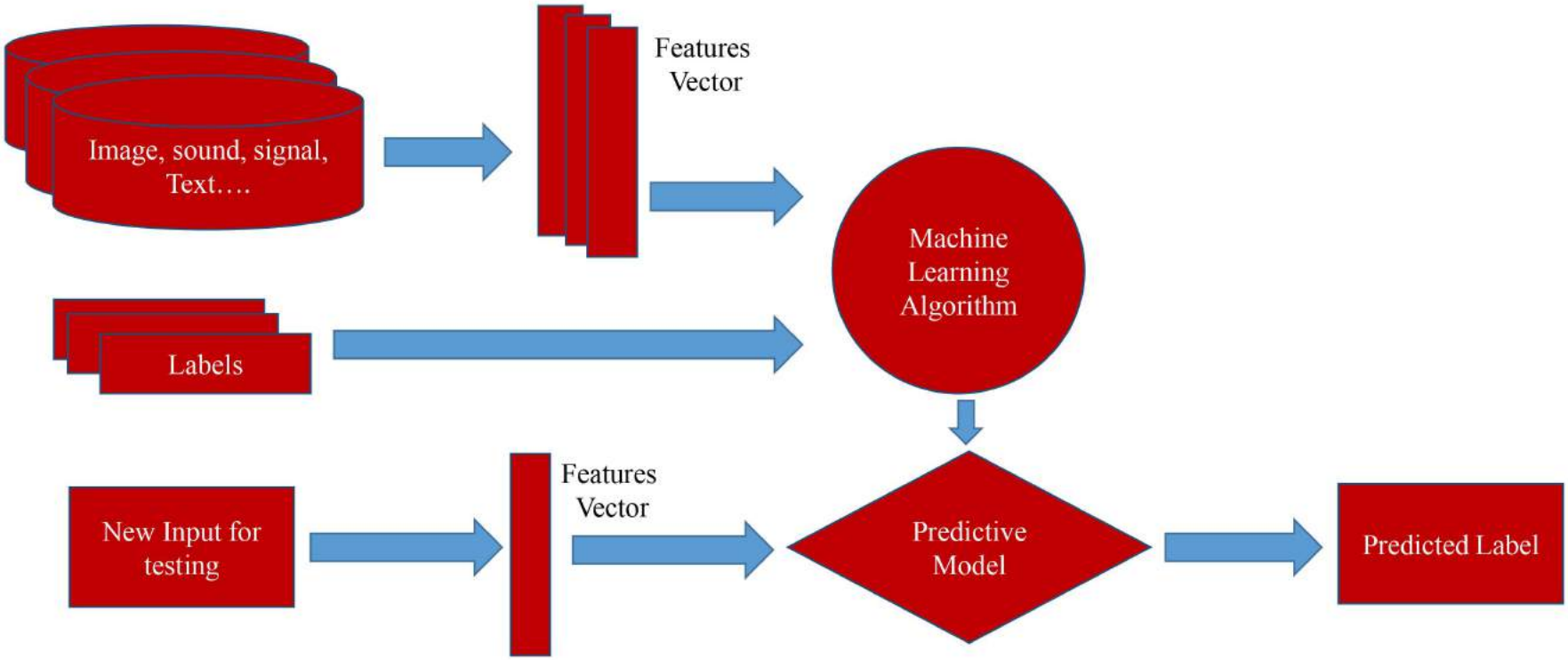
Applications of Supervised Learning



Applications of
Supervised Learning



Working flow of supervised learning



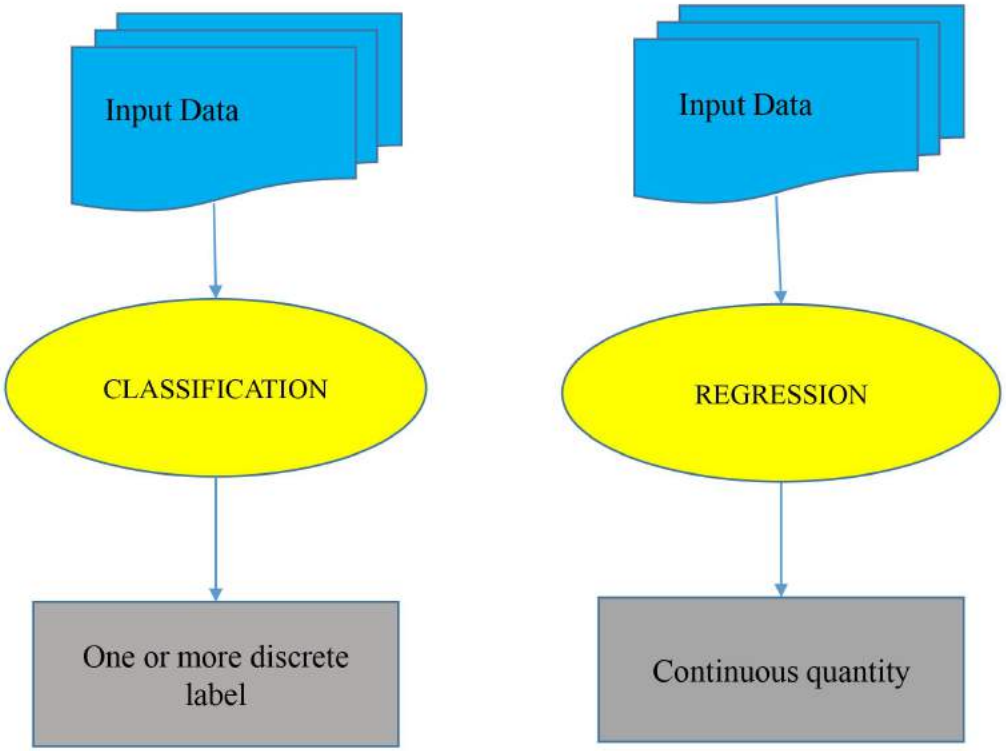
Working flow of supervised learning

- Let's examine the problem of predicting annual **income** based on the number of years of higher education someone has completed. Expressed more formally, we'd like to build a model that approximates the relationship ***f*** between the number of years of higher education **X** and corresponding annual income **Y**

$$Y = F(X) + E$$

X (input) = years of higher education
Y (output) = annual income
f = function describing the relationship between X and Y
E = random error term (positive or negative) with mean zero

Supervised Learning Techniques



Classification and Regression

Parameters	Classification	Regression
Output type	Discrete	Continuous
Trying to find	A boundary	Best Fit Line
Evaluation	Accuracy	Sum of squared errors

Examples

Will it be cold or Hot tomorrow?

What is the temperature going to be tomorrow?

42°C
(Prediction)

Classification and Regression

Students in Class

Classification

Pass or Fail?

Student Prediction

Regression

Percentage?

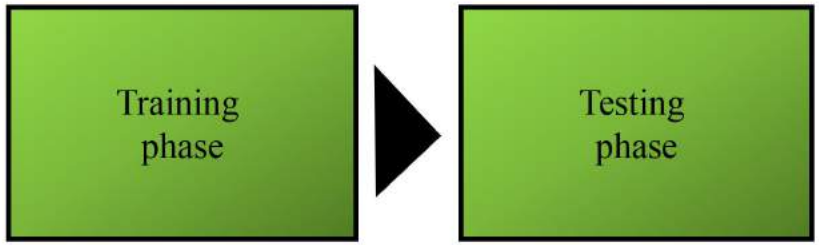
Student's marks Prediction

Classification

- Classify a document into a predefined category.
- Classification is the problem of assigning new observations to the class to which they most likely belong, based on a classification model built from labeled training data.
- More question to answer -

  Is this email spam or not?

  Is that borrower going to repay their loan?
- Some of Classification algorithms are:
  - Support Vector Machine (SVM)
  - Naïve Bayes
  - K- Nearest Neighbor
  - Decision Tree



Learning the classifier from the available data

Training set(Labeled)

Testing how well the classifier performs

Testing set

Classification Terminologies in Machine Learning

- Classifier
- Classification Model
- Feature
- Binary Classification
- Multi-Class Classification
- Multi-label Classification
- Initialize
- Train the Classifier
- Predict the Target
- Evaluate

Support Vector Machines (SVM)

Support Vector Machine

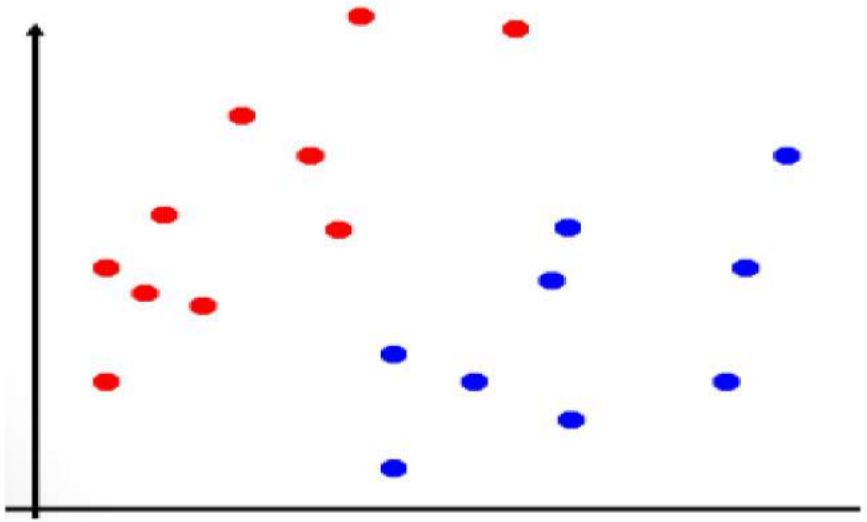
Applications

- Handwritten character/ digit recognition → Text Categorization
- Performs very well on high dimensional data.

Support Vector Machines

- "Support Vector Machine" (SVM) is a supervised machine learning algorithm which can be used for classification.
- SVMs use a separating line (or, in more than two dimensions, a multi-dimensional hyper plane) to split the space into a red zone and a blue zone.

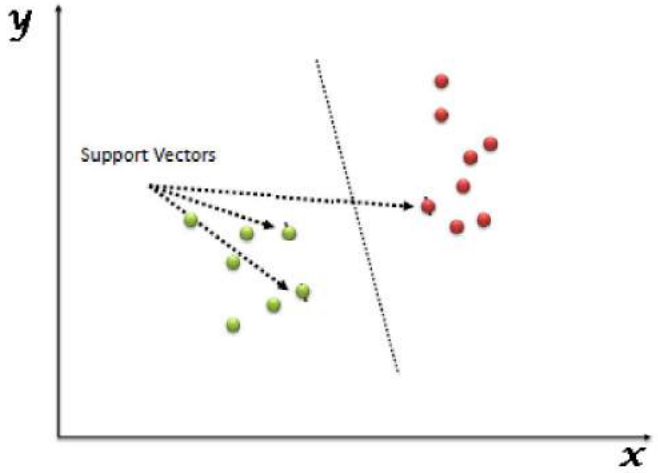
- Consider a two dimensional dataset with two classes. (Marked with red and blue circle)
- **How would we classify this dataset?**



Support Vector Machine

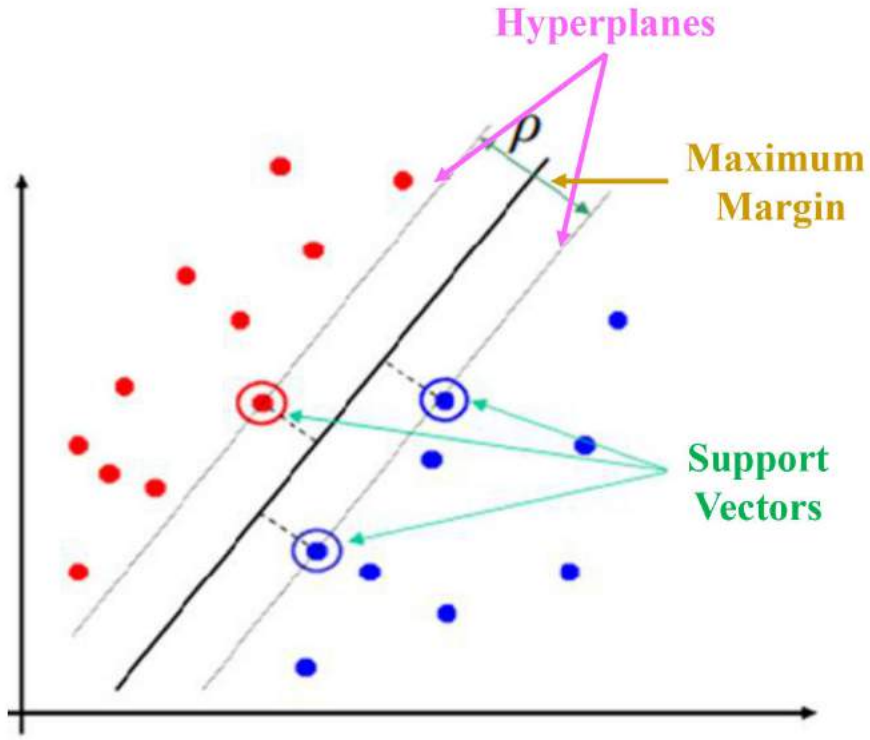
But what do you mean by Support Vector?

---- subset of training data used to represent decision boundary.



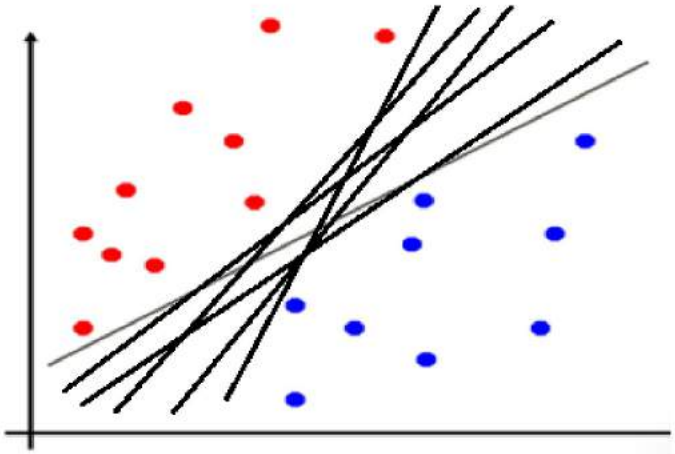
Support Vector and Hyperplanes

- Points closest to the hyperplane are support vectors.
- They are the data points most difficult to classify.
- Support Vectors have direct bearing on the optimum location of the decision surface
- The decision function is fully specified by a (usually very small) subset of training samples, the support vectors.



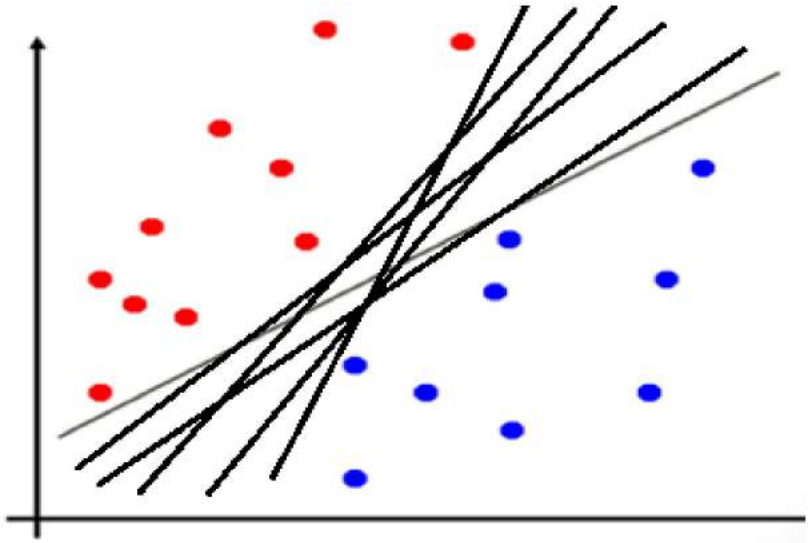
Support Vector Machine

Based on the concept of "MAXIMAL MARGIN HYPERPLANE"

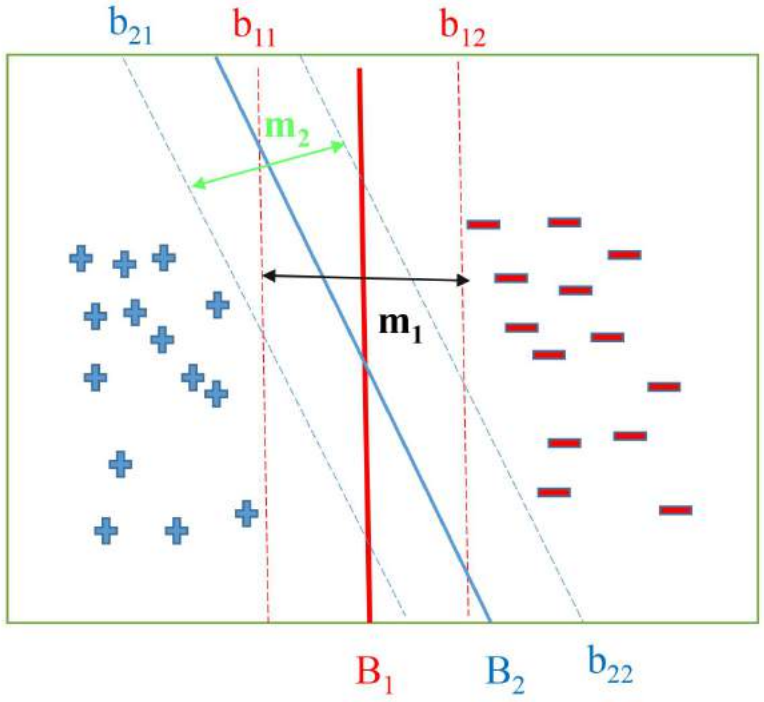


Support Vector Machine

- *Choose most optimal hyperplane.*
- *Training errors = 0*
- *Chosen Hyperplane has not equally performed well on unseen example.*



How do we choose the optimal hyperplane?



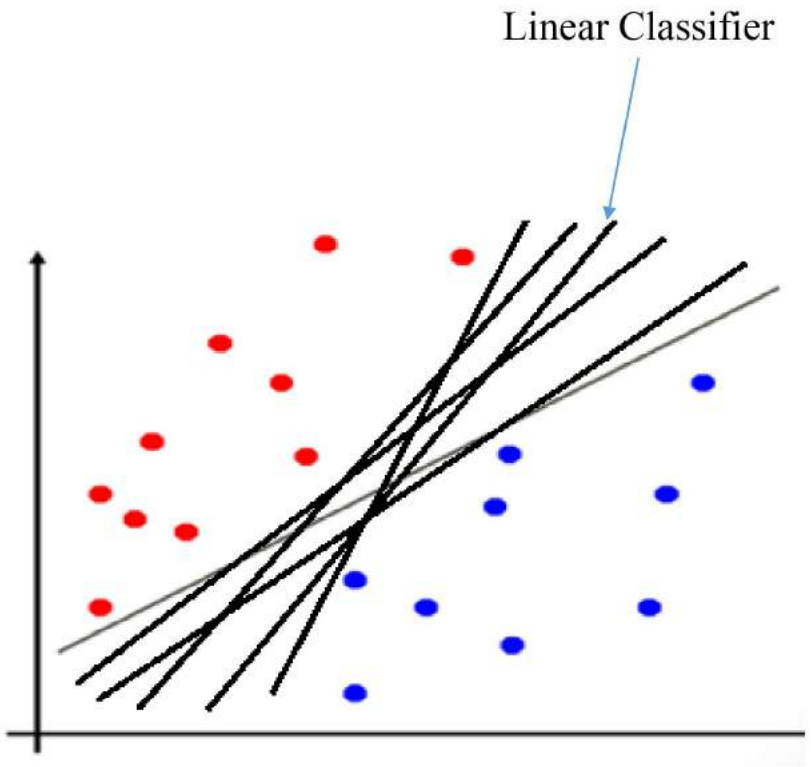
Margin(B_1) > Margin(B_2)

Linear Classifier

Any of these lines can be a classifier.

But which line is best classifier?

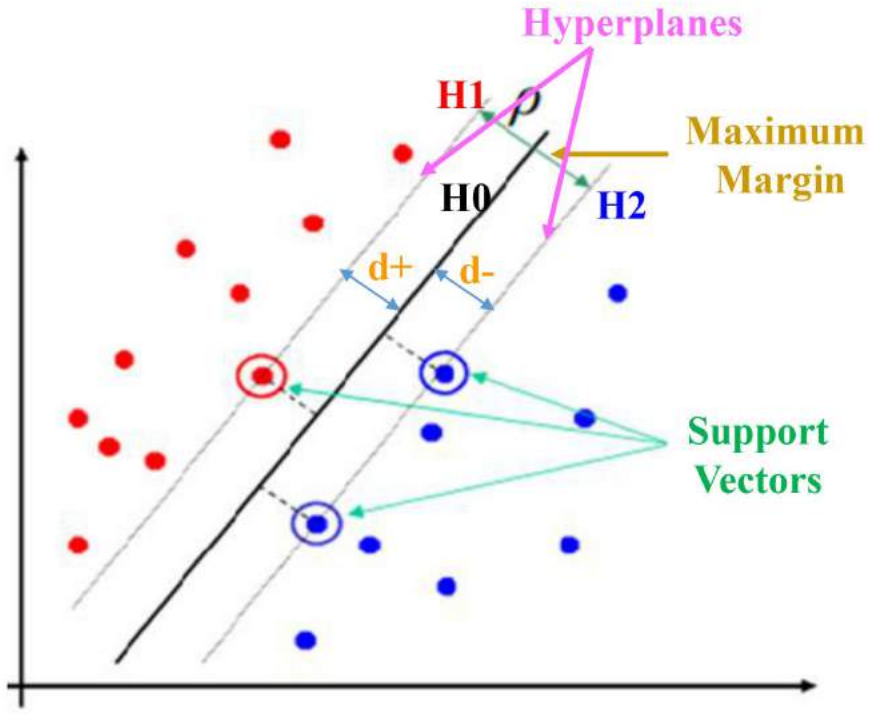
- Support Vector Machine (SVM) finds an optimal solution. Maximizes the distance between the hyperplane and the "difficult points" close to decision boundary.
- SVMs maximize the margin around the separating hyperplane.
- Which points should influence optimality? – All points?
  - Linear regression
  - Neural nets
- Only "difficult points" close to decision boundary
  - Support vector machines



Linear SVM (LSVM)- linearly separable

- A separating hyperplane (linear classifier) can be written as:
  - $w \cdot x + b = 0$

  where $w = \{w_1, w_2, \ldots, w_n\}$ is a weight vector and b a scalar (bias)
- H1 and H2 are the planes:
  - H1: $w \cdot x_i + b = +1$
  - H2: $w \cdot x_i + b = -1$
- The points on the planes H1 and H2 are the tips of the Support Vectors. The plane H_0 is the median in between, where $w \cdot x_i + b = 0$
  - d+ = the shortest distance to the closest positive point
  - d- = the shortest distance to the closest negative point.
- The margin of a separating hyperplane is (d+ + d–).



Maximal Margin Classifier (Linear SVM)

(x_i , y_i)

where i ranges from 1 to N

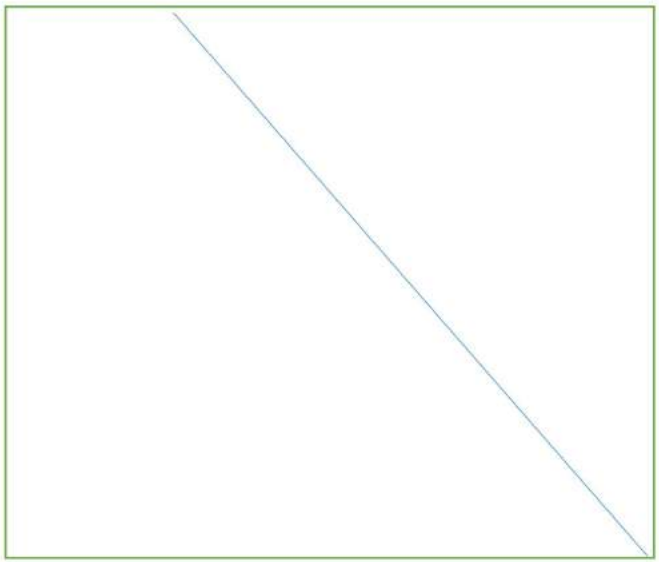
N = training instances

$y_i \in \{-1, 1\}$

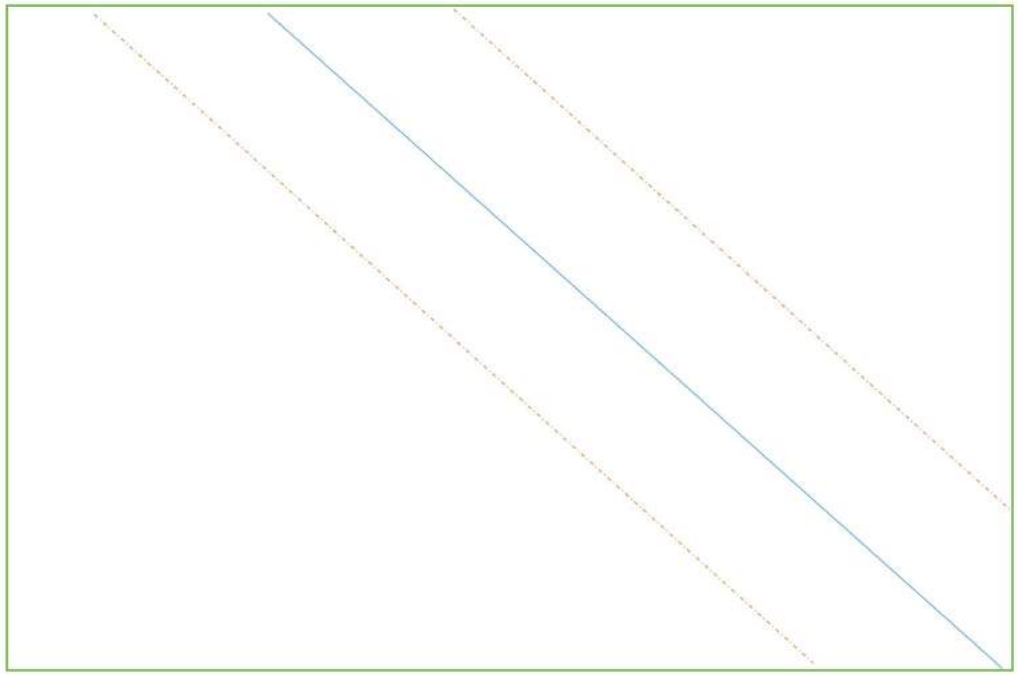
Slope – intercept form like equation:-

$y = m x + c$

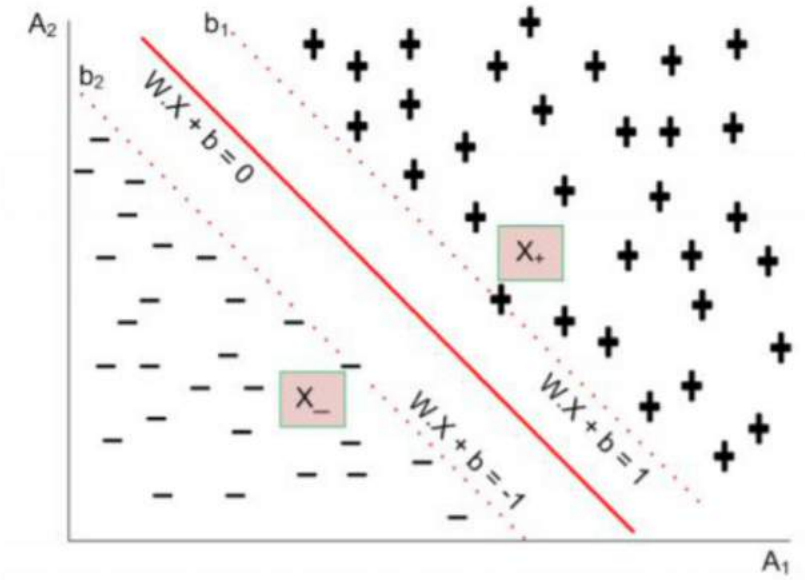
$w. x + b = 0$



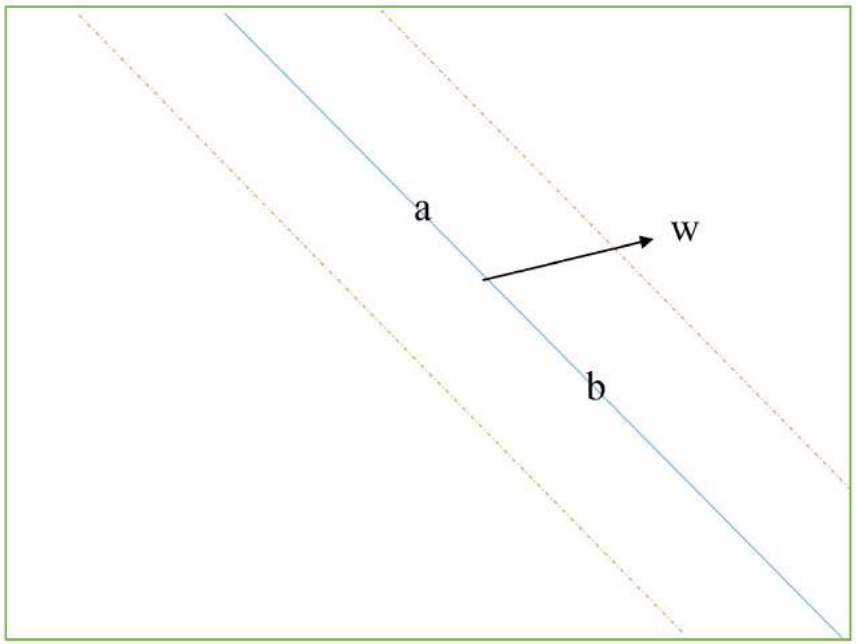
Maximal Margin Classifier (Linear SVM)



Maximal Margin Classifier (Linear SVM)



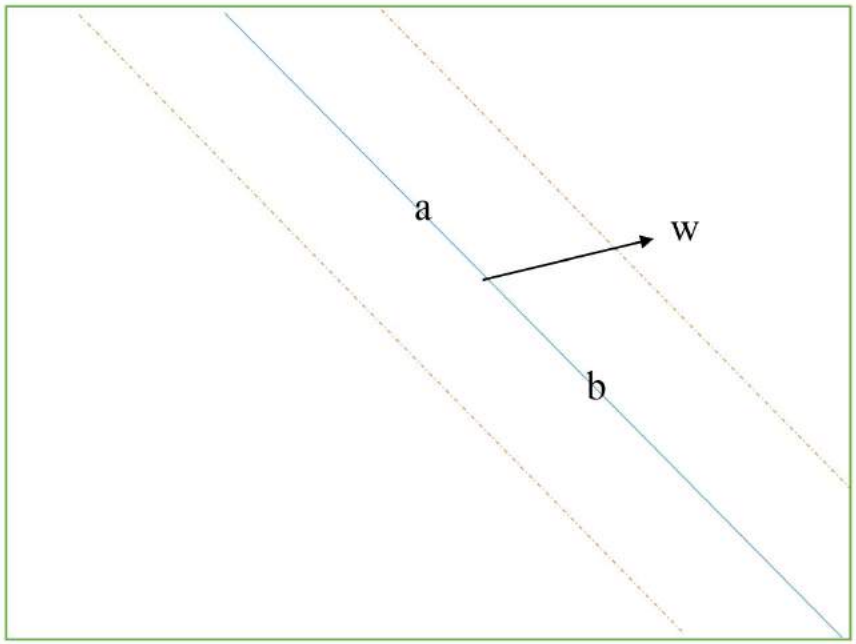
Relationship between weight vector and decision boundary



Relationship between weight vector and decision boundary

- $w.x_a + b = 0$ ……i)
- $w.x_b + b = 0$ ……ii)
- Equations i) – ii) :-
- $w\ (x_a - x_b) = 0$

W ⊥ Decision Boundary



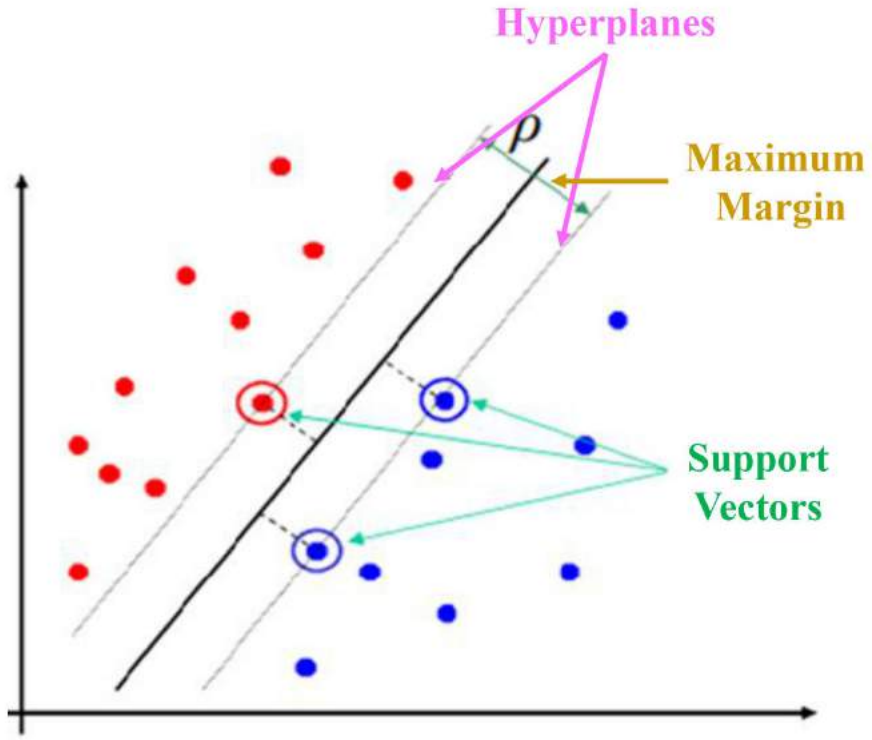
Some final definitions for SVM

- Margin of Separation (d): the separation between the hyperplane and the closest data point for a given weight vector w and bias b.

- Optimal Hyperplane (maximal margin): the particular hyperplane for which the margin of separation d is maximized.

Support Vector and Hyperplanes

- Points closest to the hyperplane are support vectors.
- They are the data points most difficult to classify.
- Support Vectors have direct bearing on the optimum location of the decision surface
- The decision function is fully specified by a (usually very small) subset of training samples, the support vectors.



Measurement of the margin of separation

- Say, variable K

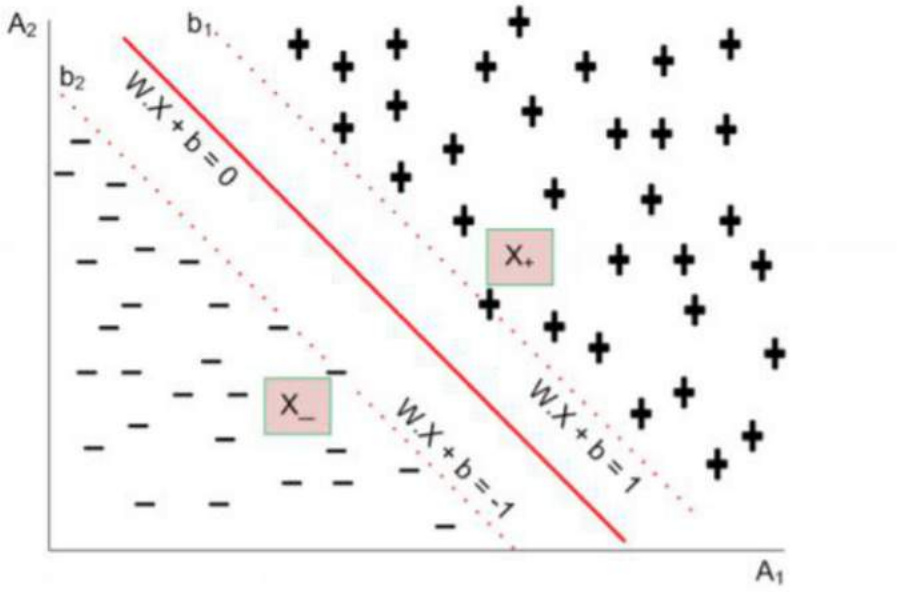
Equations:-

$wx_- + b \quad K < 0$

$wx_+ + b \quad K > 0$

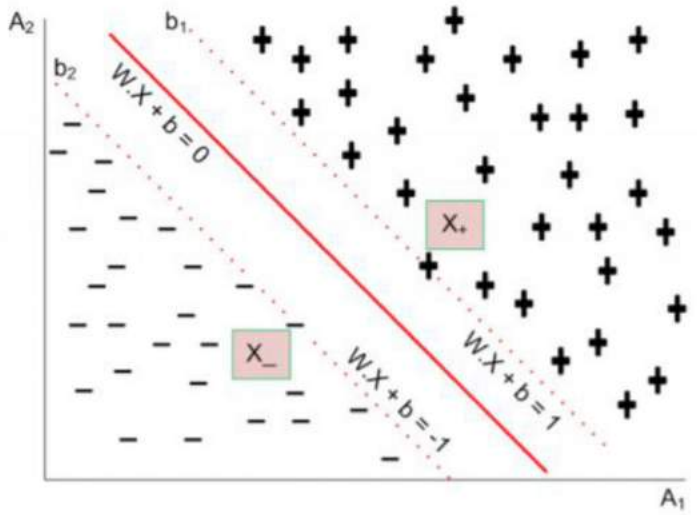
z

$$y = \begin{cases} 1 & \text{if } wz + b > 0 \\ -1 & \text{if } wz + b < 0 \end{cases}$$



b_{i1} : $w.x_1 + b = 1$ ..... i)

b_{i2} : $w.x_1 + b = -1$ .....ii)



• Eq^n i) – ii) :-

$w. (x_1 - x_2) = 1 – (-1)$

$w. (x_1 - x_2) = 2$

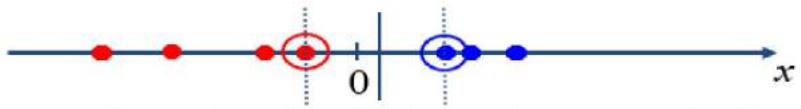
$\|w\| . d = 2$

$$d = \frac{2}{\|w\|}$$

Non-linear SVMs

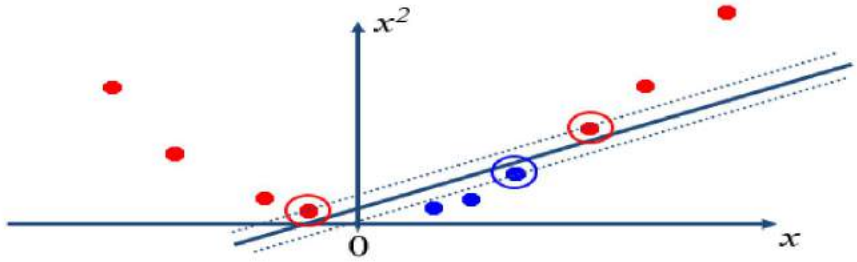
Non-linear SVMs

- Datasets that are linearly separable (with some noise) work out great:



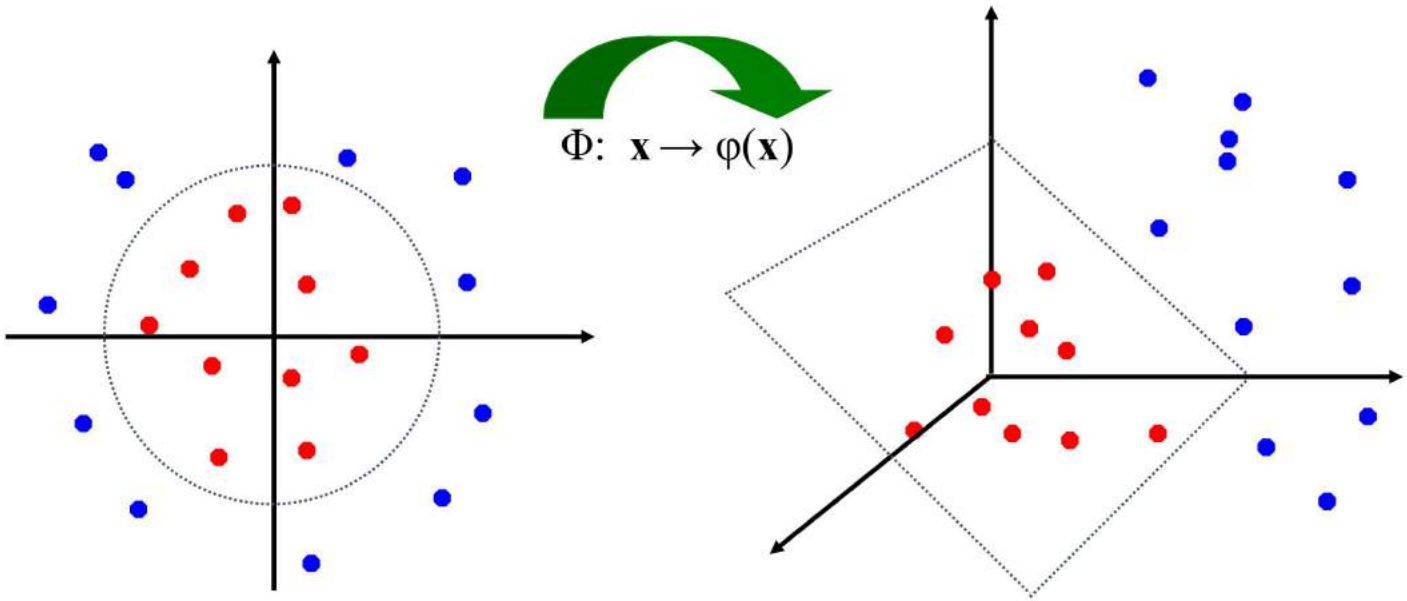
- But what are we going to do if the dataset is just too hard?

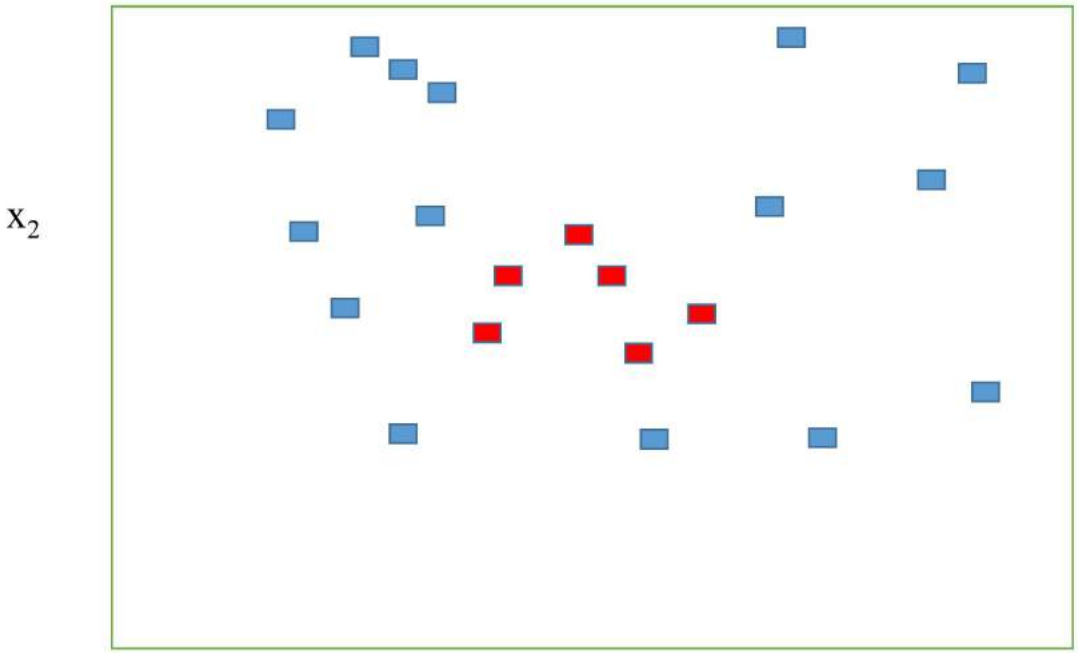
- How about…. mapping data to a higher-dimensional space:

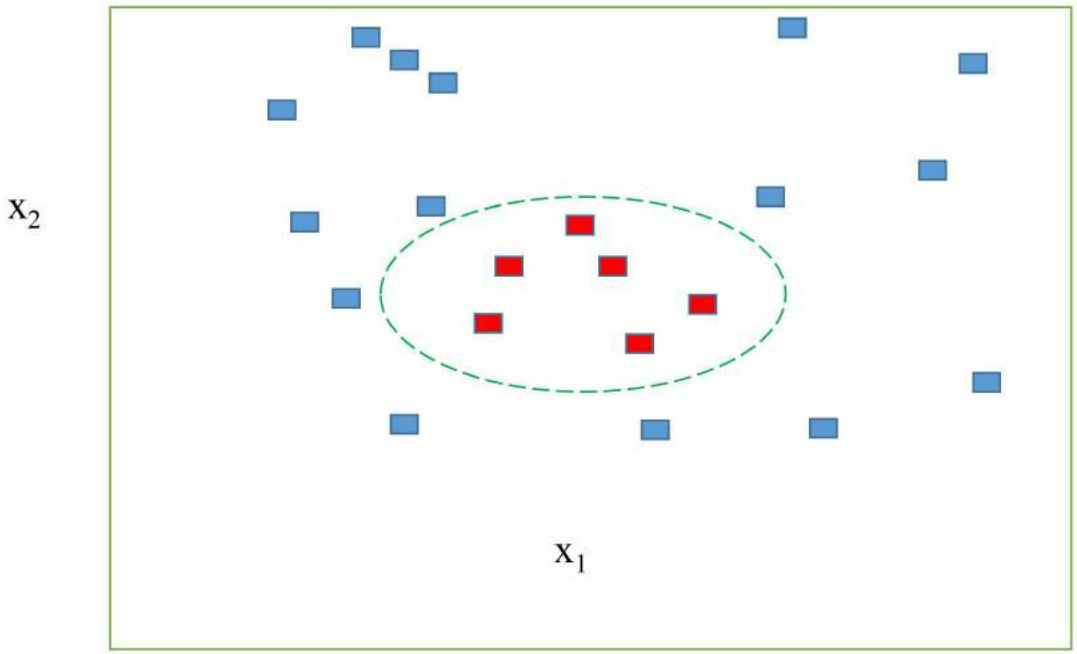


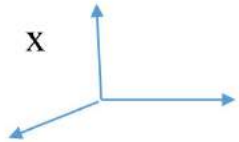
Non-linear SVM (Feature spaces)

The original feature space can always be mapped to some higher-dimensional feature space where the training set is separable:



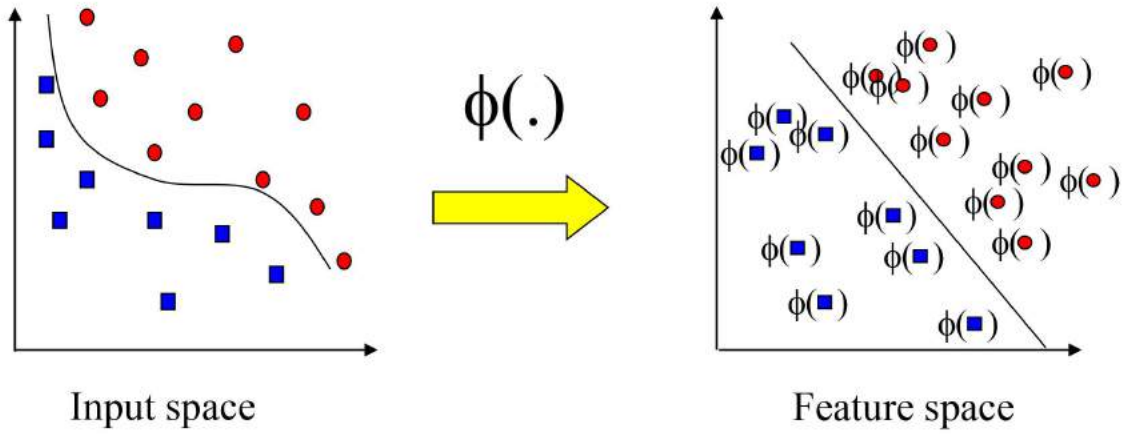

x_2
x_1


x_2
x_1


x

Transforming the data to higher dimension

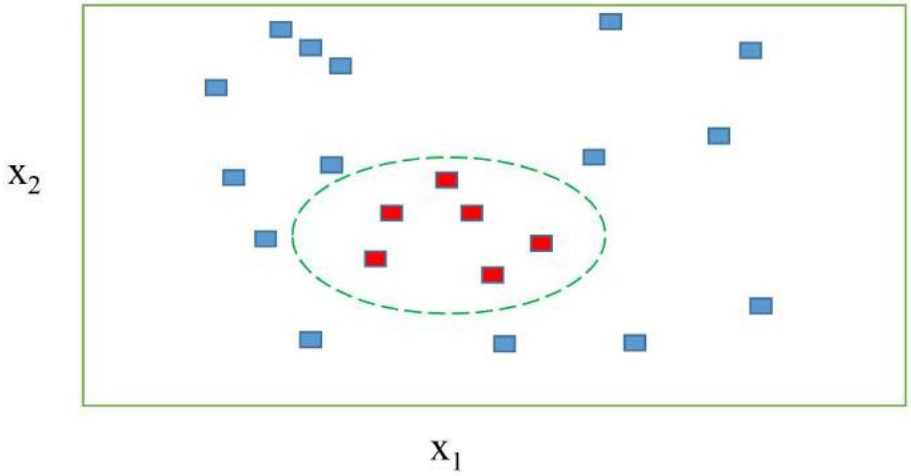
- To make it easier, transform $\mathbf{x}_i$ to a higher dimensional space:
  - Input space: the space the point $\mathbf{x}_i$ are located
  - Feature space: the space of $\boldsymbol{\varphi}(\mathbf{x}_i)$ after transformation



- $y_{(x1, x2)} = \begin{cases} 1 \\ -1 \end{cases}$

$y = 1$ if $\sqrt{(x_1 - 0.5)^2 + (x_2 - 0.5)^2} > 0.2$

$y = -1$ otherwise



$$\sqrt{(x_1 - 0.5)^2 + (x_2 - 0.5)^2} = 0.2$$

$$x_1^2 - x_1 + x_2^2 - x_2 = -0.46$$

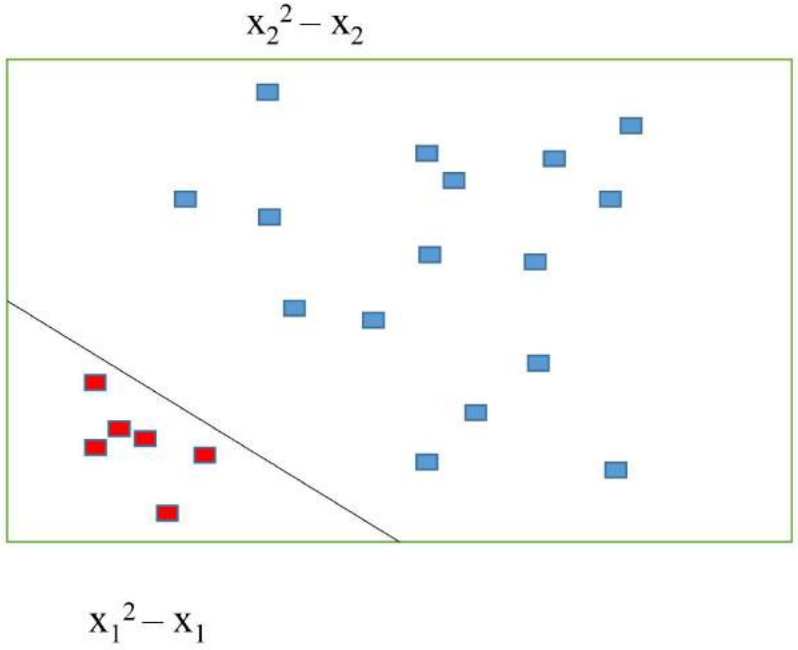
$$\emptyset : (x_1, x_2)$$

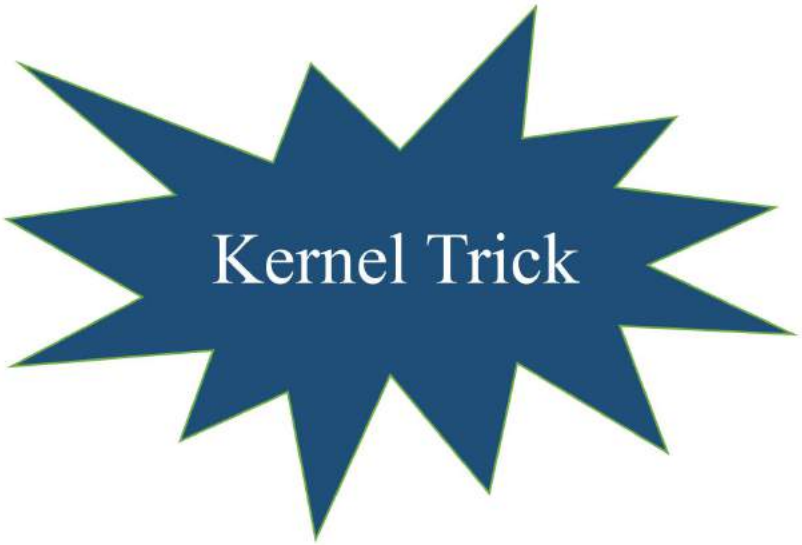
$$(x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, 1)$$

$w = w_0, w_1, \ldots\ldots w_4$

$$w_4 . x_1^2 + w_3 . x_2^2 + w_2 . \sqrt{2}x_1 + w_1 . \sqrt{2}x_2 + w_0 = 0$$

Now, the decision boundary will look like this:




Kernel Trick

Kernel Trick

- **Why should we use kernels?**
  - Make non-separable problem separable.
  - Map data into better representational space
- Common kernels
  - Linear: $K(\mathbf{x_i}, \mathbf{x_j}) = \mathbf{x_i}^T\mathbf{x_j}$
  - Polynomial of power p: $K(\mathbf{x_i}, \mathbf{x_j}) = (1+ \mathbf{x_i}^T\mathbf{x_j})^p$
  - Gaussian (radial-basis function network):

$$K(\mathbf{x_i}, \mathbf{x_j}) = \exp\left(-\frac{\|\mathbf{x_i} - \mathbf{x_j}\|^2}{2\sigma^2}\right)$$

  - Sigmoid (Neural Network Activation Function): $K(\mathbf{x_i}, \mathbf{x_j}) = \tanh(\beta_0 \mathbf{x_i}^T\mathbf{x_j} + \beta_1)$

$$( ) \cdot ( )$$

vector | Another vector

$$\emptyset(x_1) \cdot \emptyset(x_2)$$

$$\emptyset(u) \cdot \emptyset(v)$$

$$(u_1^2 , u_2^2 , \sqrt{2}u_1, \sqrt{2}u_2, 1) \cdot (v_1^2 , v_2^2 , \sqrt{2}v_1, \sqrt{2}v_2, 1)$$

$$(u_1^2 , u_2^2 , \sqrt{2}u_1, \sqrt{2}u_2, 1) \cdot (v_1^2 , v_2^2 , \sqrt{2}v_1, \sqrt{2}v_2, 1)$$

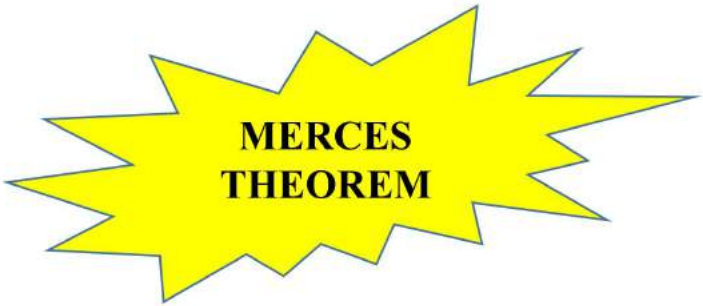
Dot product (multiply) :

$$u_1^2 v_1^2 + u_2^2 v_2^2 + 2 u_1 v_1 + 2 u_2 v_2 + 1$$

$$(u \cdot v + 1)^2$$

That is $(a + b)^2$

Kernel function



K(u, v)

K(u, v) ----- $\emptyset(u)$. $\emptyset(v)$

Transforming the data to higher dimension

- If every data point is mapped into high-dimensional space via some transformation Φ: $\mathbf{x} \rightarrow \varphi(\mathbf{x})$, the inner product becomes:

$$\mathbf{K(x_i, x_j)= \varphi(x_i)^T \varphi(x_j)}$$

- A kernel function is some function that corresponds to an inner product in some expanded feature space. Example:

  2-dimensional vectors $\mathbf{x}=[x_1 \ x_2]$; let $K(\mathbf{x_i}, \mathbf{x_j}) = (1 + \mathbf{x_i}^T\mathbf{x_j})^2$,

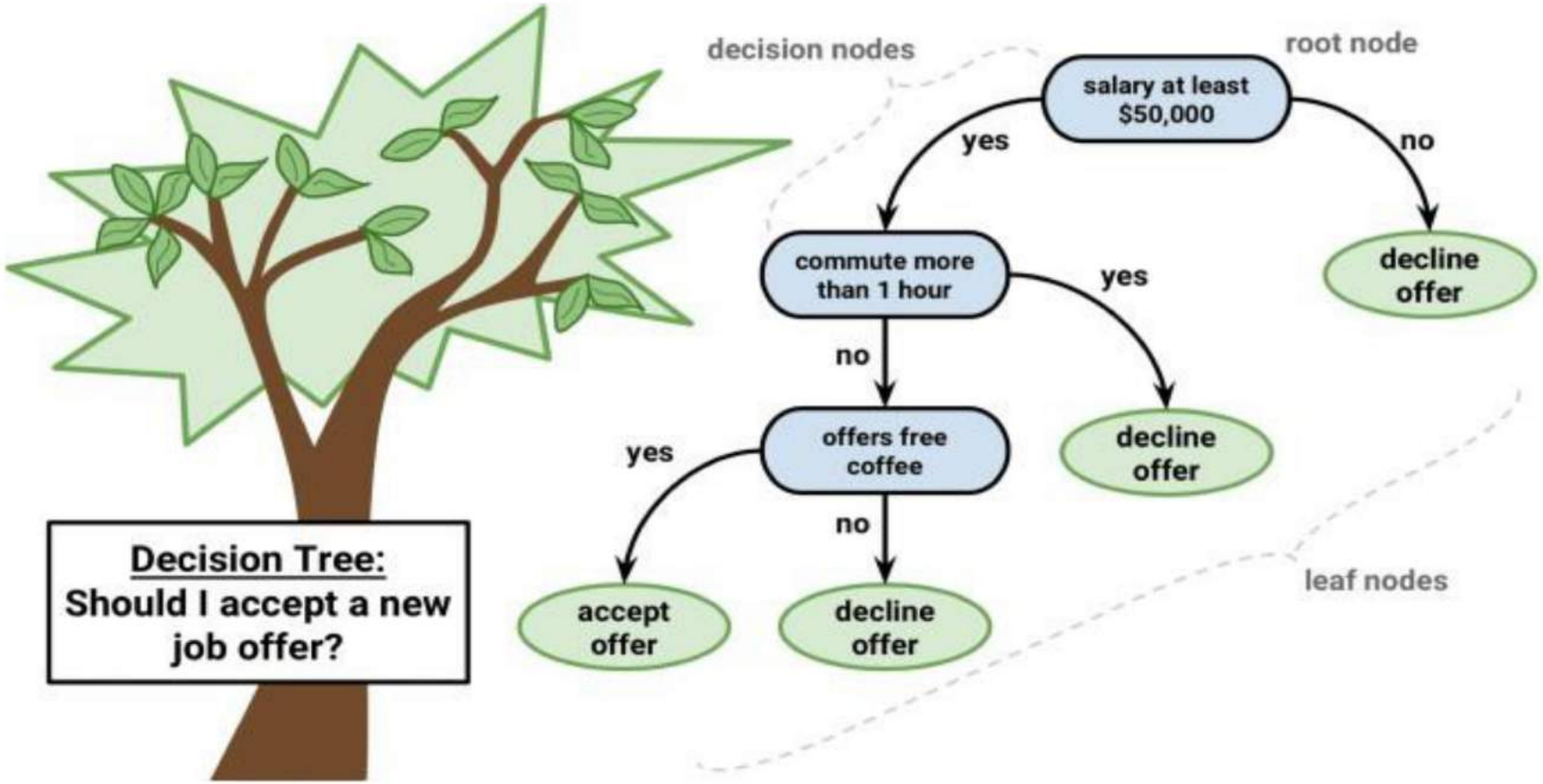
  Need to show that $K(\mathbf{x_i}, \mathbf{x_j}) = \varphi(\mathbf{x_i})^T\varphi(\mathbf{x_j})$:

$$K(\mathbf{x_i},\mathbf{x_j}) = (1 + \mathbf{x_i}^T\mathbf{x_j})^2 = 1+ x_{i1}^2x_{j1}^2 + 2\, x_{i1}x_{j1}\, x_{i2}x_{j2} + x_{i2}^2x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2}$$

$$= [1 \ \ x_{i1}^2 \ \sqrt{2}\, x_{i1}x_{i2} \ \ x_{i2}^2 \ \sqrt{2}x_{i1} \ \sqrt{2}x_{i2}]^T \, [1 \ \ x_{j1}^2 \ \sqrt{2}\, x_{j1}x_{j2} \ \ x_{j2}^2 \ \sqrt{2}x_{j1} \ \sqrt{2}x_{j2}]$$

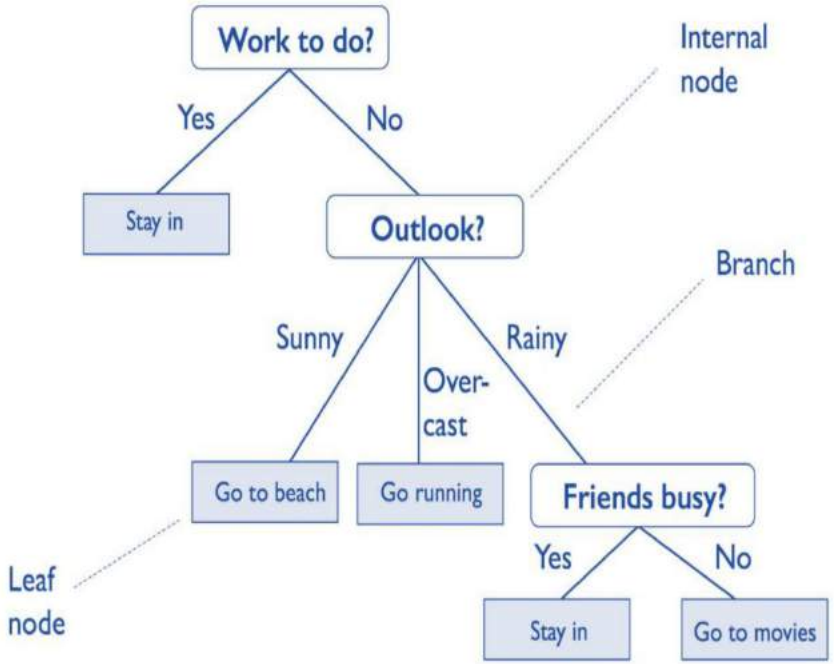
$$= \varphi(\mathbf{x_i})^T\varphi(\mathbf{x_j}) \quad \text{where } \varphi(\mathbf{x}) = [1 \ \ x_1^2 \ \sqrt{2}\, x_1x_2 \ \ x_2^2 \ \sqrt{2}x_1 \ \sqrt{2}x_2]$$

Decision Trees

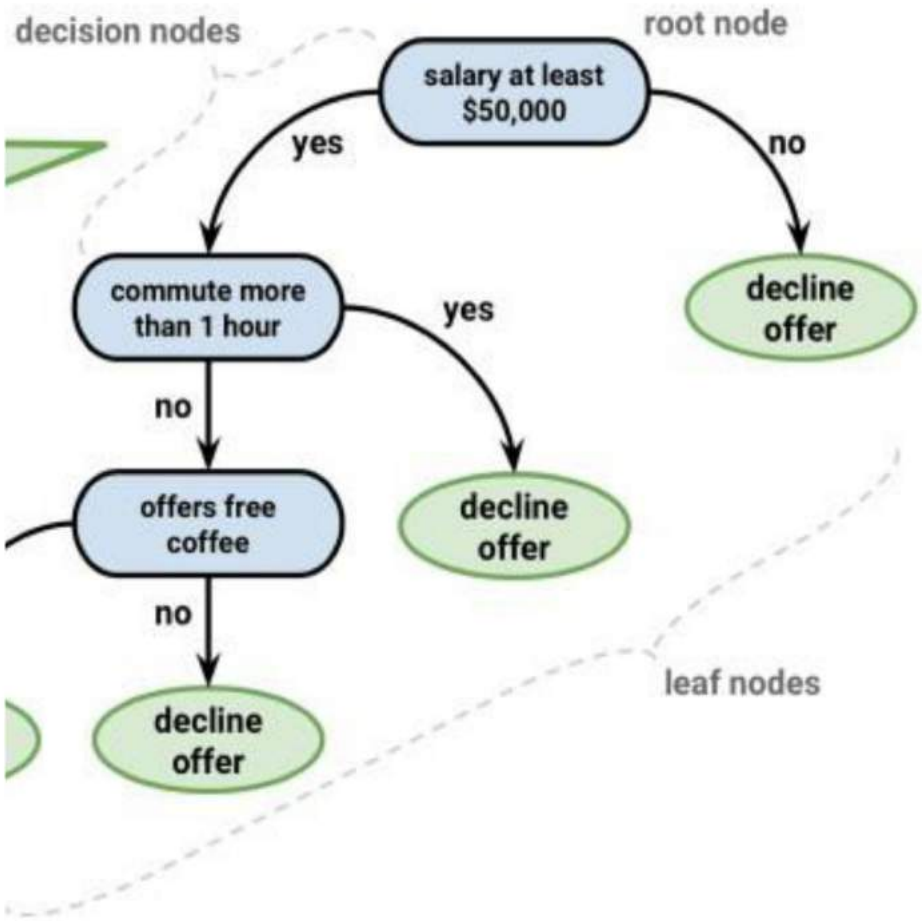

Decision Tree:
Should I accept a new job offer?
decision nodes
root node
salary at least $50,000
yes
no
commute more than 1 hour
yes
decline offer
no
offers free coffee
yes
decline offer
yes
no
accept offer
decline offer
leaf nodes

Decision Tree

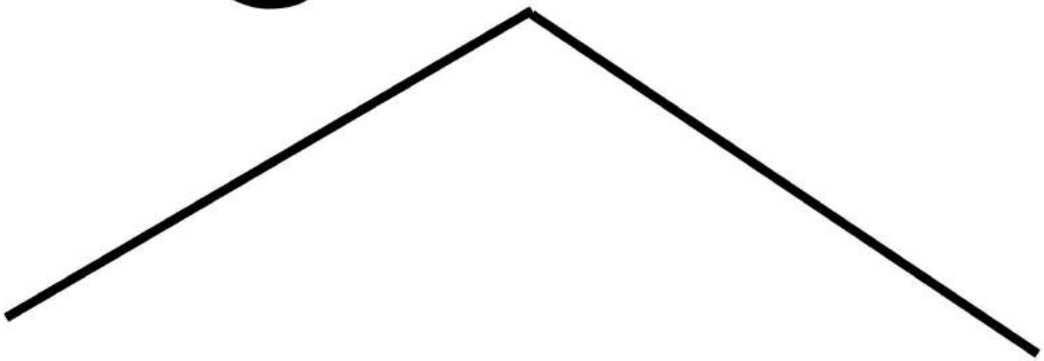
- Decision tree learning is one of the most widely used techniques for classification. It builds classification or regression models in the form of a tree structure.
  - Its classification accuracy is competitive with other methods.
  - It is very efficient.
- It breaks down a dataset into smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with **decision nodes** and **leaf nodes**.
- Non-linear classifier
- Easy to use and easy to interpret



A Decision Tree is a tree where each node represents the Feature (Attribute), each link (Branch) represents a decision (rule) and each leaf represents an outcome.



Algorithms



Cart

- Gini Index

ID3

- Entropy Function
- Information Gain

S. No.	Outlook	Temperature	Humidity	Windy	PlayTennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rainy	Mild	High	Weak	Yes
5	Rainy	Cool	Normal	Weak	Yes
6	Rainy	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rainy	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rainy	Mild	High	Strong	No

Make a decision tree that predicts whether tennis will be played on the day?

S. No.	Outlook	Temperature	Humidity	Windy	PlayTennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rainy	Mild	High	Weak	Yes
5	Rainy	Cool	Normal	Weak	Yes
6	Rainy	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rainy	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rainy	Mild	High	Strong	No

Step 1: Create a root node

- How to choose the root node?

The attribute that best classifies the training data, use this attribute at the root of the tree.

- How to choose the Best attribute?

So from here, *ID3 algorithm* **begins**

- In Decision Tree the major challenge is to identification of the attribute for the root node in each level. This process is known as attribute selection. The core algorithm for building decision trees called **ID3** which employs a top-down, greedy search through the space of possible branches with no backtracking.

- We have two popular attribute selection measures ID3 uses:
  - **Entropy** - ID3 algorithm uses entropy to calculate the homogeneity of a sample. If the sample is completely homogeneous the entropy is zero and if the sample is an equally divided it has entropy of one.
  - **Information Gain** - The information gain is based on the decrease in entropy after a dataset is split on an attribute. Constructing a decision tree is all about finding attribute that returns the highest information gain

- Calculate **Entropy** (Amount of uncertainity in dataset):

$$Entropy = \frac{-p}{p+n} log_2(\frac{p}{p+n}) - \frac{n}{p+n} log_2(\frac{n}{p+n})$$

- Calculate **Average Information**:

$$I(Attribute) = \sum \frac{p_i + n_i}{p+n} Entropy(A)$$

- Calculate **Information Gain**: (Difference in Entropy before and after splitting dataset on attribute A)

$$Gain = Entropy(S) - I(Attribute)$$

1. Compute the **entropy** for data-set **Entropy(s)**

2. for every attribute/feature:

    1. calculate entropy for all other values **Entropy(a)**

    2. take **Average information entropy** for the current attribute

    3. calculate **gain** for the current attribute

    4. pick the **highest gain attribute**.

5. **Repeat** Until we get the tree we desired.

1.

S. No.	Outlook	Temperature	Humidity	Windy	PlayTennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rainy	Mild	High	Weak	Yes
5	Rainy	Cool	Normal	Weak	Yes
6	Rainy	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rainy	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rainy	Mild	High	Strong	No

P = 9

N = 5

Total = 14

- Calculate **Entropy(S):**

$$Entropy = \frac{-p}{p+n} log_2(\frac{p}{p+n}) - \frac{n}{p+n} log_2(\frac{n}{p+n})$$

$$Entropy(S) = \frac{-9}{9+5} log_2(\frac{9}{9+5}) - \frac{5}{9+5} log_2(\frac{5}{9+5})$$

$$Entropy(S) = \frac{-9}{14} log_2(\frac{9}{14}) - \frac{5}{14} log_2(\frac{5}{14}) \quad = 0.940$$

- For each Attribute: (let say **Outlook**)
  - Calculate Entropy for each Values, i.e for 'Sunny', 'Rainy','Overcast'

Outlook	PlayTennis
Sunny	No
Sunny	No
Sunny	No
Sunny	Yes
Sunny	Yes

Outlook	PlayTennis
Rainy	Yes
Rainy	Yes
Rainy	No
Rainy	Yes
Rainy	No

Outlook	PlayTennis
Overcast	Yes
Overcast	Yes
Overcast	Yes
Overcast	Yes

Outlook	p	n	Entropy
Sunny	2	3	0.971
Rainy	3	2	0.971
Overcast	4	0	0

Calculate **Entropy(Outlook='Value'):**

$$Entropy = \frac{-p}{p+n} log_2(\frac{p}{p+n}) - \frac{n}{p+n} log_2(\frac{n}{p+n})$$

$$\text{E (Outlook=sunny)} = -\frac{2}{5}\log\left(\frac{2}{5}\right) - \frac{3}{5}\log\left(\frac{3}{5}\right) = 0.971$$

$$\text{E (Outlook=overcast)} = -1\log(1) - 0\log(0) = 0$$

$$\text{E (Outlook=rainy)} = -\frac{3}{5}\log\left(\frac{3}{5}\right) - \frac{2}{5}\log\left(\frac{2}{5}\right) = 0.971$$

- Calculate **Average Information Entropy**:

$$I(Outlook) = \frac{p_{sunny} + n_{sunny}}{p+n} Entropy(Outlook = Sunny) +$$

$$\frac{p_{rainy} + n_{rainy}}{p+n} Entropy(Outlook = Rainy) +$$

$$\frac{p_{Overcast} + n_{Overcast}}{p+n} Entropy(Outlook = Overcast)$$

$$I(Outlook) = \frac{3+2}{9+5} * 0.971 + \frac{2+3}{9+5} * 0.971 + \frac{4+0}{9+5} * 0 \quad = 0.693$$

- Calculate **Gain**: attribute is Outlook

$$Gain = Entropy(S) - I(Attribute)$$

$$Entropy(S) = 0.940$$

$$Gain(Outlook) = 0.940 - 0.693 = 0.247$$

- For each Attribute: (let say **Temperature**)
  - Calculate Entropy for each Temp, i.e for 'Hot', 'Mild' and 'Cool'

Temperature	PlayTennis
Hot	No
Hot	No
Hot	Yes
Hot	Yes

Temperature	PlayTennis
Mild	Yes
Mild	No
Mild	Yes
Mild	Yes
Mild	Yes
Mild	No

Temperature	PlayTennis
Cool	Yes
Cool	No
Cool	Yes
Cool	Yes

Temperature	p	n	Entropy
Hot	2	2	1
Mild	4	2	0.918
Cool	3	1	0.811

- Calculate **Average Information Entropy**:

$$I(Temperature) = \frac{p_{hot} + n_{hot}}{p+n} Entropy(Temperature = Hot) +$$

$$\frac{p_{mild} + n_{mild}}{p+n} Entropy(Temperature = Mild) +$$

$$\frac{p_{Cool} + n_{Cool}}{p+n} Entropy(Temperature = Cool)$$

$$I(Temperature) = \frac{2+2}{9+5} * 1 + \frac{4+2}{9+5} * 0.918 + \frac{3+1}{9+5} * 0.811 => 0.911$$

- Calculate **Gain**: attribute is Temperature

$$Gain = Entropy(S) - I(Attribute)$$

$$Entropy(S) = 0.940$$

$$Gain(Temperature) = 0.940 - 0.911 = 0.029$$

- For each Attribute: (let say **Humidity**)
  - Calculate Entropy for each Humidity, i.e for 'High', 'Normal'

Humidity	PlayTennis
Normal	Yes
Normal	No
Normal	Yes
Normal	Yes
Normal	Yes
Normal	Yes
Normal	Yes

Humidity	PlayTennis
High	No
High	No
High	Yes
High	Yes
High	No
High	Yes
High	No

Humidity	p	n	Entropy
High	3	4	0.985
Normal	6	1	0.591

- Calculate **Average Information Entropy**:

$$I(Humidity) = \frac{p_{High} + n_{High}}{p+n} Entropy(Humidity = High) +$$

$$\frac{p_{Normal} + n_{Normal}}{p+n} Entropy(Humidity = Normal)$$

$$I(Humidity) = \frac{3+4}{9+5} * 0.985 + \frac{6+1}{9+5} * 0.591 => 0.788$$

- Calculate **Gain**: attribute is Humidity

$$Gain = Entropy(S) - I(Attribute)$$

$$Entropy(S) = 0.940$$

$$Gain(Humidity) = 0.940 - 0.788 = 0.152$$

- For each Attribute: (let say **Windy**)
  - Calculate Entropy for each Windy, i.e for 'Strong' and 'Weak'

Windy	PlayTennis
Weak	No
Weak	Yes
Weak	Yes
Weak	Yes
Weak	No
Weak	Yes
Weak	Yes
Weak	Yes

Windy	PlayTennis
Strong	No
Strong	No
Strong	Yes
Strong	Yes
Strong	Yes
Strong	No

Windy	p	n	Entropy
Strong	3	3	1
Weak	6	2	0.811

- Calculate **Average Information Entropy**:

$$I(Windy) = \frac{p_{Strong} + n_{Strong}}{p+n} Entropy(Windy = Strong) +$$

$$\frac{p_{Weak} + n_{Weak}}{p+n} Entropy(Windy = Weak)$$

$$I(Windy) = \frac{3+3}{9+5} * 1 + \frac{6+2}{9+5} * 0.811 => 0.892$$

- Calculate **Gain**: attribute is Windy

$$Gain = Entropy(S) - I(Attribute)$$

$$Entropy(S) = 0.940$$

$$Gain(Windy) = 0.940 - 0.892 = 0.048$$

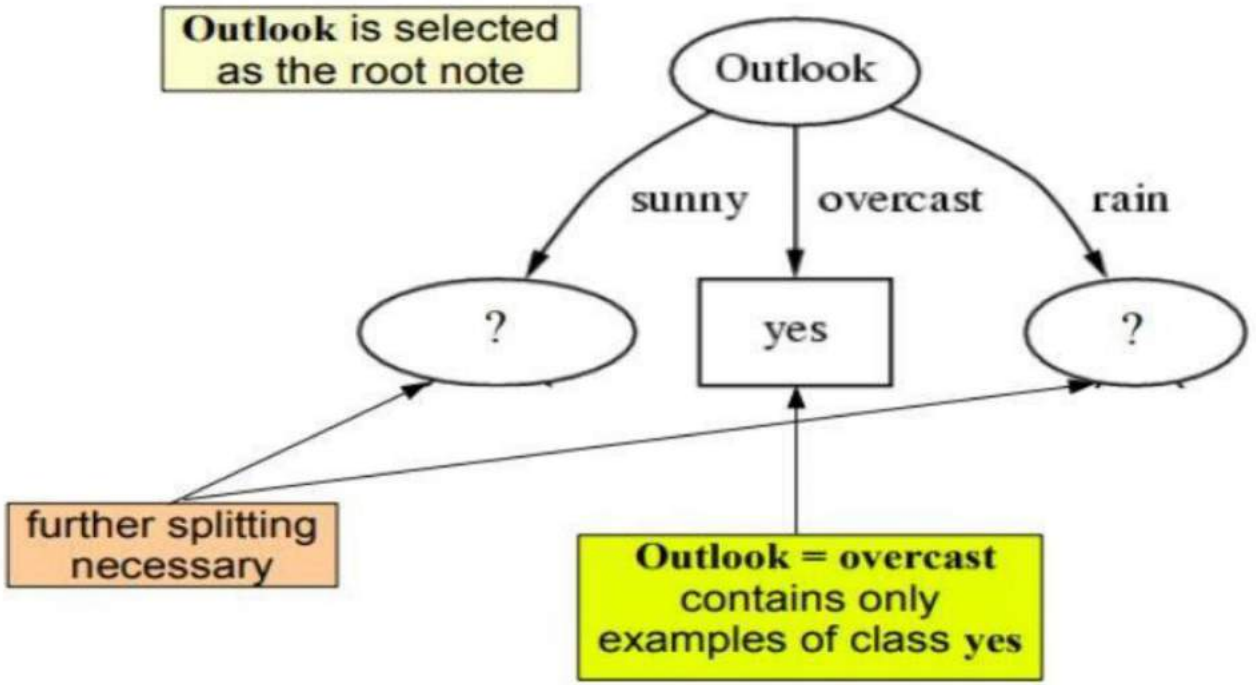
Pick the highest gain attribute.

Attributes	Gain
Outlook	0.247
Temperature	0.029
Humidity	0.152
Windy	0.048

Root Node:

OUTLOOK

Outlook	Temperature	Humidity	Windy	PlayTennis
Overcast	Hot	High	Weak	Yes
Overcast	Cool	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes



- Repeat the same thing for sUB-trees till we get the tree.

Outlook	Temperature	Humidity	Windy	PlayTennis
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes

Outlook = "SuNNy"

Outlook	Temperature	Humidity	Windy	PlayTennis
Rainy	Mild	High	Weak	Yes
Rainy	Cool	Normal	Weak	Yes
Rainy	Cool	Normal	Strong	No
Rainy	Mild	Normal	Weak	Yes
Rainy	Mild	High	Strong	No

Outlook = "RaiNy"

Outlook	Temperature	Humidity	Windy	PlayTennis
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes

P= 2 N= 3 Total= 5

- Entropy:

$$Entropy = \frac{-p}{p+n} log_2(\frac{p}{p+n}) - \frac{n}{p+n} log_2(\frac{n}{p+n})$$

$$Entropy(S_{sunny}) = \frac{-2}{2+3} log_2(\frac{2}{2+3}) - \frac{3}{2+3} log_2(\frac{3}{2+3})$$

=>0.971

- For each Attribute: (let say **Humidity**):
  - Calculate Entropy for each Humidity, i.e for 'High' and 'Normal'

Outlook	Humidity	PlayTennis
Sunny	High	No
Sunny	High	No
Sunny	High	No
Sunny	Normal	Yes
Sunny	Normal	Yes

Humidity	p	n	Entropy
high	0	3	0
normal	2	0	0

  - Calculate **Average Information Entropy**: I(Humidity) = 0
  - Calculate **Gain:** Gain = 0.971

- For each Attribute: (let say **Windy**):
  - Calculate Entropy for each Windy, i.e for 'Strong' and 'Weak'

Outlook	Windy	PlayTennis
Sunny	Strong	No
Sunny	Strong	Yes
Sunny	Weak	No
Sunny	Weak	No
Sunny	Weak	Yes

Windy	p	n	Entropy
Strong	1	1	1
Weak	1	2	0.918

  - Calculate **Average Information Entropy**: I(Windy) = 0.951
  - Calculate **Gain:** Gain = 0.020

- For each Attribute: (let say **Temperature**):
  - Calculate Entropy for each Windy, i.e for 'Cool', 'Hot' and 'Mild'

Outlook	Temperature	PlayTennis
Sunny	Cool	Yes
Sunny	Hot	No
Sunny	Hot	No
Sunny	Mild	No
Sunny	Mild	Yes

Temperature	p	n	Entropy
Cool	1	0	0
Hot	0	2	0
Mild	1	1	1

  - Calculate **Average Information Entropy**: I(Temp) = 0.4
  - Calculate **Gain:** Gain = 0.571

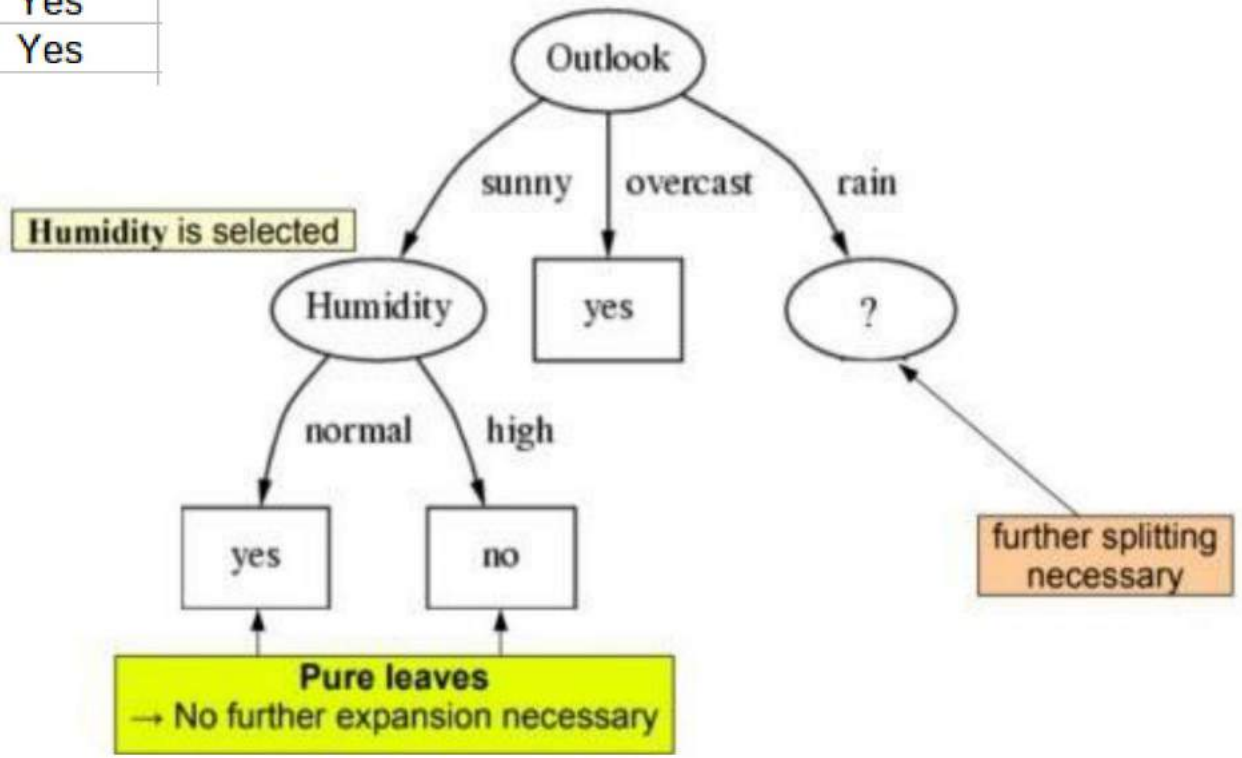
Pick the highest gain attribute.

Attributes	Gain
Temperature	0.571
Humidity	0.971
Windy	0.02

Next Node in sunny:

Humidity

Outlook	Humidity	PlayTennis
Sunny	High	No
Sunny	High	No
Sunny	High	No
Sunny	Normal	Yes
Sunny	Normal	Yes



Outlook	Temperature	Humidity	Windy	PlayTennis
Rainy	Mild	High	Weak	Yes
Rainy	Cool	Normal	Weak	Yes
Rainy	Cool	Normal	Strong	No
Rainy	Mild	Normal	Weak	Yes
Rainy	Mild	High	Strong	No

P= 3

N= 2

Total = 5

- Entropy:

$$Entropy = \frac{-p}{p+n} log_2(\frac{p}{p+n}) - \frac{n}{p+n} log_2(\frac{n}{p+n})$$

$$Entropy(S_{Rainy}) = \frac{-3}{3+2} log_2(\frac{3}{3+2}) - \frac{2}{3+2} log_2(\frac{2}{2+3})$$

=>0.971

- For each Attribute: (let say **Humidity**):
  - Calculate Entropy for each Humidity, i.e for 'High' and 'Normal'

Outlook	Humidity	PlayTennis
Rainy	High	Yes
Rainy	High	No
Rainy	Normal	Yes
Rainy	Normal	No
Rainy	Normal	Yes

Attribute	p	n	Entropy
High	1	1	1
Normal	2	1	0.918

  - Calculate **Average Information Entropy**: I(Humidity) = 0.951
  - Calculate **Gain:** Gain = 0.020

- For each Attribute: (let say **Windy**):
  - Calculate Entropy for each Windy, i.e for 'Strong' and 'Weak'

Outlook	Windy	PlayTennis
Rainy	Strong	No
Rainy	Strong	No
Rainy	Weak	Yes
Rainy	Weak	Yes
Rainy	Weak	Yes

Attribute	p	n	Entropy
Strong	0	2	0
Weak	3	0	0

- Calculate **Average Information Entropy**: I(Windy) = 0
- Calculate **Gain:** Gain = 0.971

- For each Attribute: (let say **Temperature**):
  - Calculate Entropy for each Windy, i.e for 'Cool', 'Hot' and 'Mild'

Outlook	Temperature	PlayTennis
Rainy	Mild	Yes
Rainy	Cool	Yes
Rainy	Cool	No
Rainy	Mild	Yes
Rainy	Mild	No

Attribute	p	n	Entropy
Cool	1	1	1
Mild	2	1	0.918

  - Calculate **Average Information Entropy**: I(Temp) = 0.951
  - Calculate **Gain:** Gain = 0.020

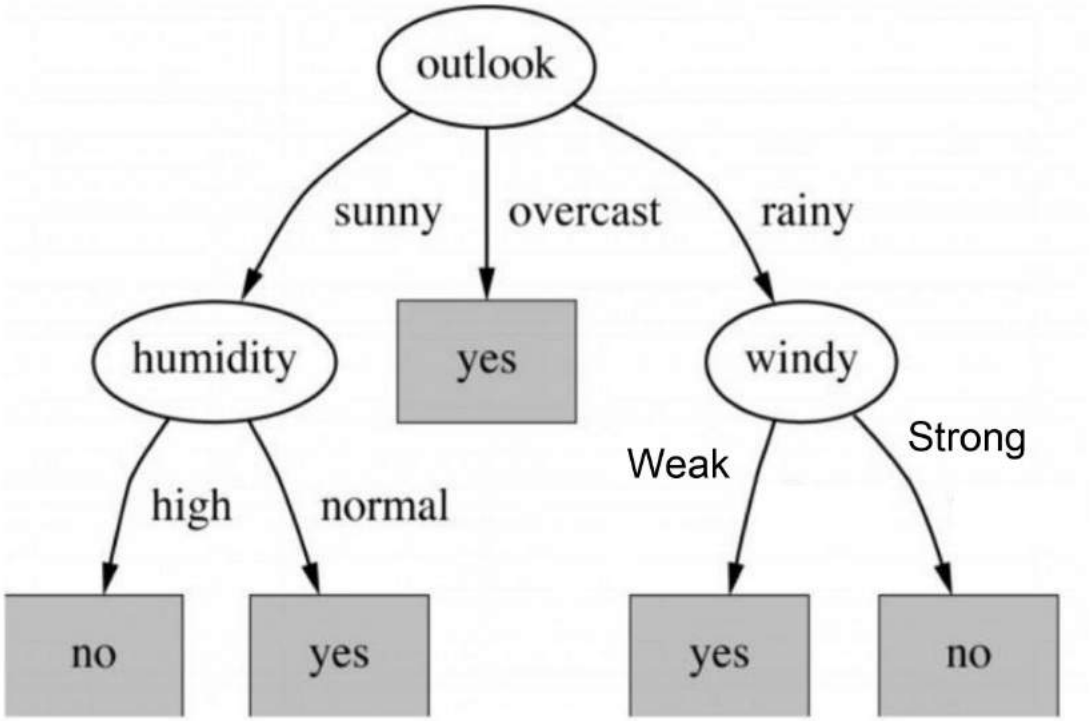
- Pick the highest gain attribute.

Attributes	Gain
Humidity	0.02
Windy	0.971
Temperature	0.02

Next Node in Rainy:

Windy

Final decision tree



Decision Tree to Decision Rules

- A decision tree can easily be transformed to a set of rules by mapping from the root node to the leaf nodes one by one.

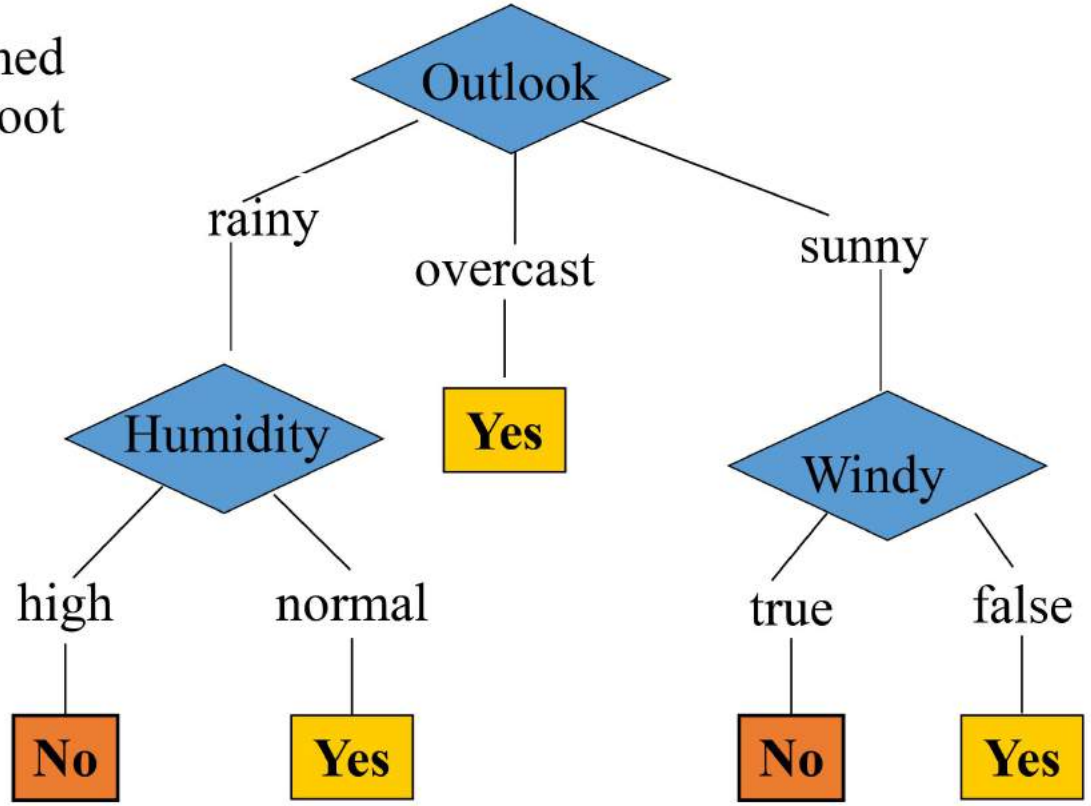
R_1: IF (Outlook=Sunny) AND (Windy=FALSE) THEN Play=Yes

R_2: IF (Outlook=Sunny) AND (Windy=TRUE) THEN Play=No

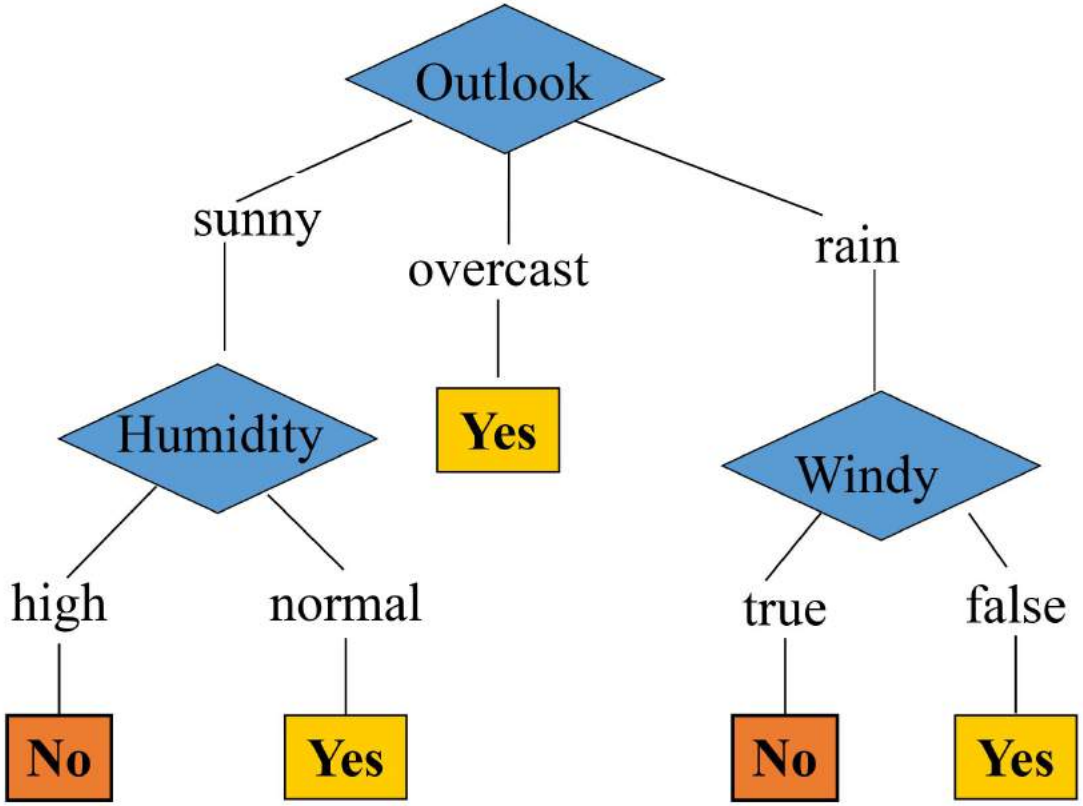
R_3: IF (Outlook=Overcast) THEN Play=Yes

R_4: IF (Outlook=Rainy) AND (Humidity=High) THEN Play=No

R_5: IF (Outlook=Rain) AND (Humidity=Normal) THEN Play=Yes



To ‘**play tennis**’ or not!



A new test example:
(Outlook==rain) and
(Windy==false)

Pass it on the tree
-> Decision is yes.

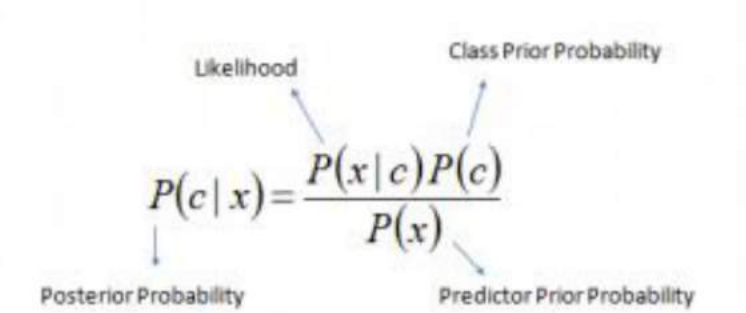
Decision Trees - Issues

- Working with continuous attributes (binning)
- Avoid Overfitting - several approaches to avoiding overfitting in building decision trees.
  - **Pre-pruning** that stop growing the tree earlier, before it perfectly classifies the training set.
  - **Post-pruning** that allows the tree to perfectly classify the training set, and then post prune the tree.
- Super attributes (attributes with many unique values)
- Working with missing values

The Naïve Bayes Model

The Naïve Bayes Model

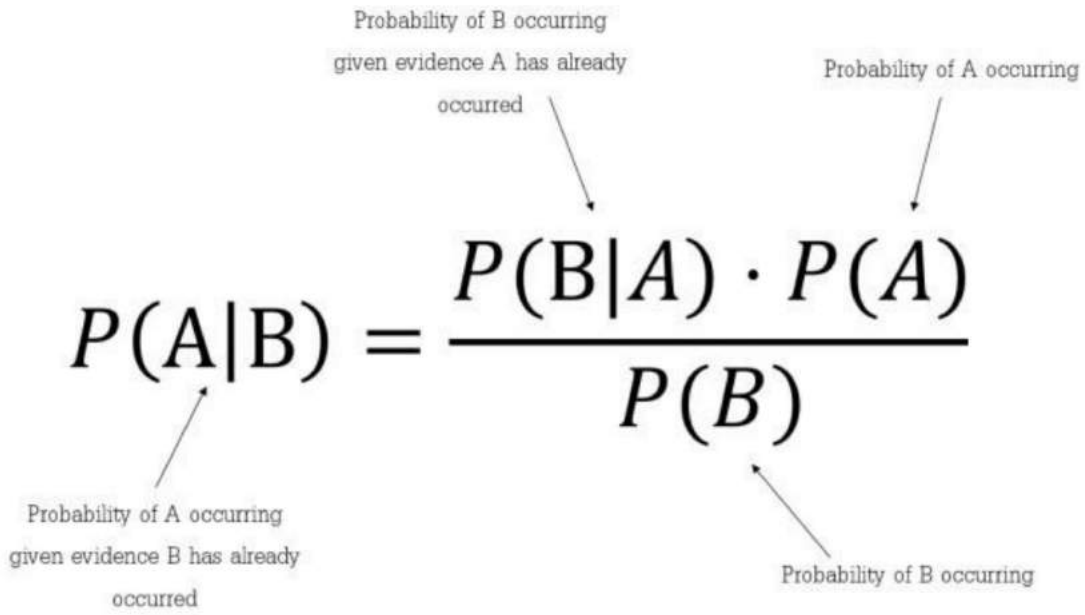
- In Naïve Bayes classifier all features are independent of given the class label Y(predictors). Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature
- Naive Bayes classifiers are a collection of classification algorithms based on **Bayes' Theorem**.
- Bayes theorem provides a way of calculating posterior probability P(c|x) from P(c), P(x) and P(x|c). Look at the equation below:



$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

- P(c|x) is the posterior probability of class (c, target) given predictor (x, attributes).
- P(c) is the prior probability of class.
- P(x|c) is the likelihood which is the probability of predictor given class.
- P(x) is the prior probability of predictor.

The Mathematics of Naïve Bayes Model



Continued

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

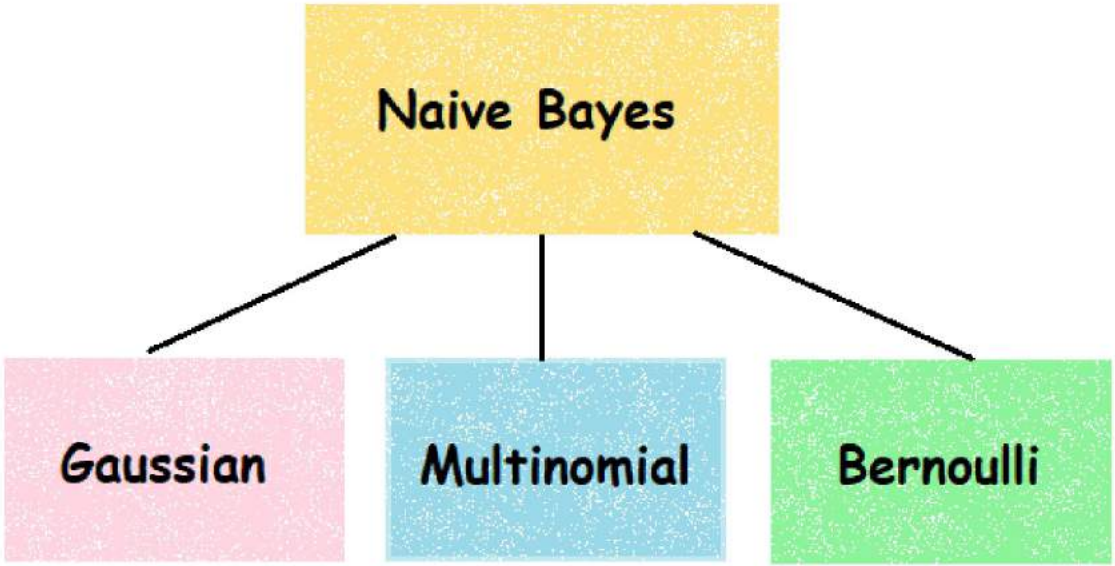
where A and B are events and $P(B) \neq 0$.

- $P(A \mid B)$ is a conditional probability: the likelihood of event A occurring given that B is true.
- $P(B \mid A)$ is also a conditional probability: the likelihood of event B occurring given that A is true.
- $P(A)$ and $P(B)$ are the probabilities of observing A and B independently of each other; this is known as the marginal probability.

$$\text{Posterior} = \frac{\text{Likelihood} * \text{Prior}}{\text{Evidence}}$$

$$P(target \mid predictor) = \frac{P(predictor \mid target) \cdot P(target)}{P(predictor)}$$

Types of Naïve Bayes Algorithm



The Naïve Bayes Model

- Naïve Bayes Algorithm: Assumption that all input attributes are conditionally independent!
- Learning Phase: Given a training set **S**,

For each target value of c_i $(c_i = c_1, \cdots, c_L)$

$\hat{P}(C = c_i) \leftarrow$ estimate $P(C = c_i)$ with examples in **S**;

For every attribute value x_{jk} of each attribute X_j $(j = 1, \cdots, n; k = 1, \cdots, N_j)$

$\hat{P}(X_j = x_{jk} \mid C = c_i) \leftarrow$ estimate $P(X_j = x_{jk} \mid C = c_i)$ with examples in **S**;

Output: conditional probability tables;

Test Phase: Given an unknown instance $\mathbf{X}' = (a_1', \cdots, a_n')$

Look up tables to assign the label c^* to **X'** if

$$[\hat{P}(a_1' \mid c^*) \cdots \hat{P}(a_n' \mid c^*)]\hat{P}(c^*) > [\hat{P}(a_1' \mid c) \cdots \hat{P}(a_n' \mid c)]\hat{P}(c), \quad c \neq c^*, c = c_1, \cdots, c_L$$

Example of the Naïve Bayes Classifier

Training

The weather data, with counts and probabilities												
outlook			temperature			humidity			windy		play	
	yes	no		yes	no		yes	no		yes	no	yes	no
sunny	2	3	hot	2	2	high	3	4	false	6	2	9	5
overcast	4	0	mild	4	2	normal	6	1	true	3	3		
rainy	3	2	cool	3	1								
sunny	2/9	3/5	hot	2/9	2/5	high	3/9	4/5	false	6/9	2/5	9/14	5/14
overcast	4/9	0/5	mild	4/9	2/5	normal	6/9	1/5	true	3/9	3/5		
rainy	3/9	2/5	cool	3/9	1/5								

Query:

A new day				
outlook	temperature	humidity	windy	play
sunny	cool	high	true	?

Example

- Learning Phase

Outlook	Play=*Yes*	Play=*No*
Sunny	2/9	3/5
Overcast	4/9	0/5
Rain	3/9	2/5

Temperature	Play=*Yes*	Play=*No*
Hot	2/9	2/5
Mild	4/9	2/5
Cool	3/9	1/5

Humidity	Play=*Yes*	Play=*No*
High	3/9	4/5
Normal	6/9	1/5

Wind	Play=*Yes*	Play=*No*
Strong	3/9	3/5
Weak	6/9	2/5

$P(\text{Play}=Yes) = 9/14 \quad P(\text{Play}=No) = 5/14$

Example

- Test Phase
  - Given a new instance,

    $\mathbf{x}'$=(Outlook=*Sunny*, Temperature=*Cool*, Humidity=*High*, Wind=*Strong*)

  - Look up tables

P(Outlook=*Sunny* | Play=*Yes*) = 2/9
P(Temperature=*Cool* | Play=*Yes*) = 3/9
P(Huminity=*High* | Play=*Yes*) = 3/9
P(Wind=*Strong* | Play=*Yes*) = 3/9
P(Play=*Yes*) = 9/14

P(Outlook=*Sunny* | Play=*No*) = 3/5
P(Temperature=*Cool* | Play==*No*) = 1/5
P(Huminity=*High* | Play=*No*) = 4/5
P(Wind=*Strong* | Play=*No*) = 3/5
P(Play=*No*) = 5/14

MAP: **M**aximum **A** **P**osterior Rule

- Likelihood of yes

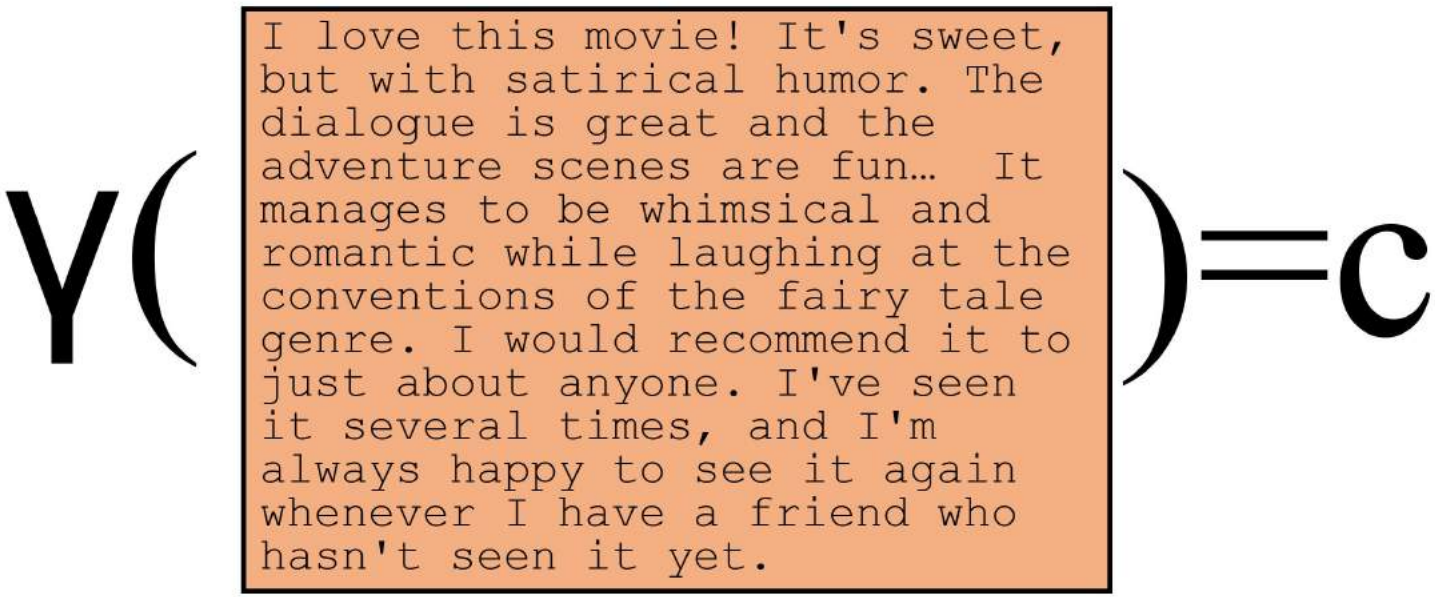
$$= \frac{2}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{9}{14} = 0.0053$$

- Likelihood of no

$$= \frac{3}{5} \times \frac{1}{5} \times \frac{4}{5} \times \frac{3}{5} \times \frac{5}{14} = 0.0206$$

- Given the fact $P(Yes \mid \mathbf{X}') < P(No \mid \mathbf{X}')$, we label $\mathbf{X}'$ to be "*No*".

Text Classification(Example)

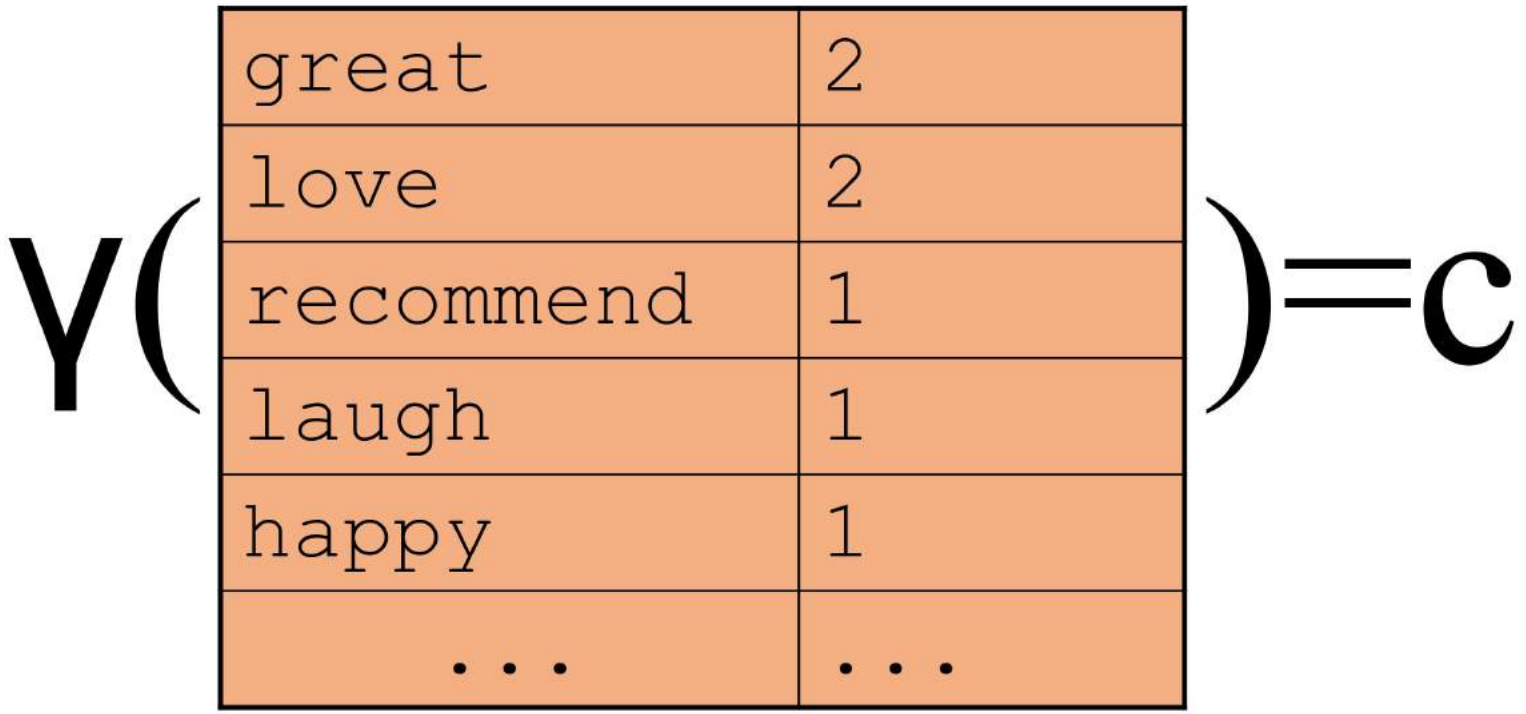


Text Classification(Example)

Text Classification(Example)

The bag of words representation: using a subset of words

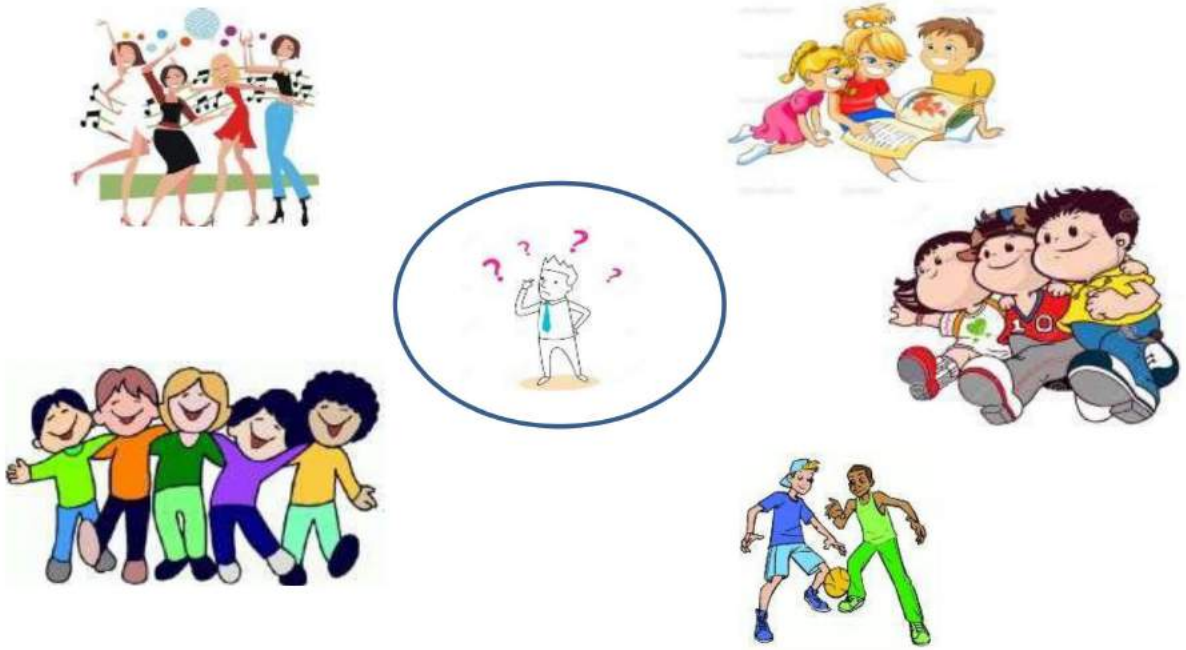
Text Classification(Example)



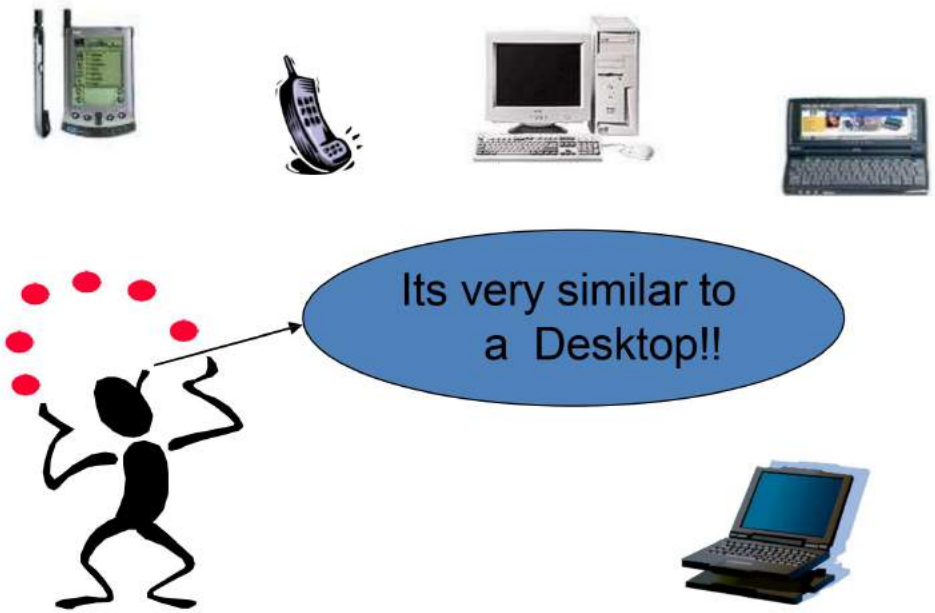
KNN Classifier

KNN Classifier : Simple analogy

- Tell me about your friends(*who your neighbors are*) and *I will tell you who you are*.



Instance-based Learning



Principle of KNN:-

….“Birds of the same feather flock together.”

Other names of this algorithm:-

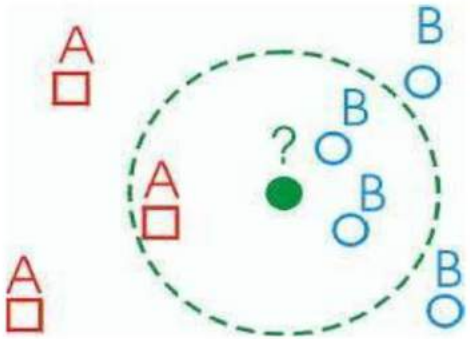
- K-Nearest Neighbors
- Memory-Based Reasoning
- Example-Based Reasoning
- Instance-Based Learning
- Lazy Learning

What is KNN?

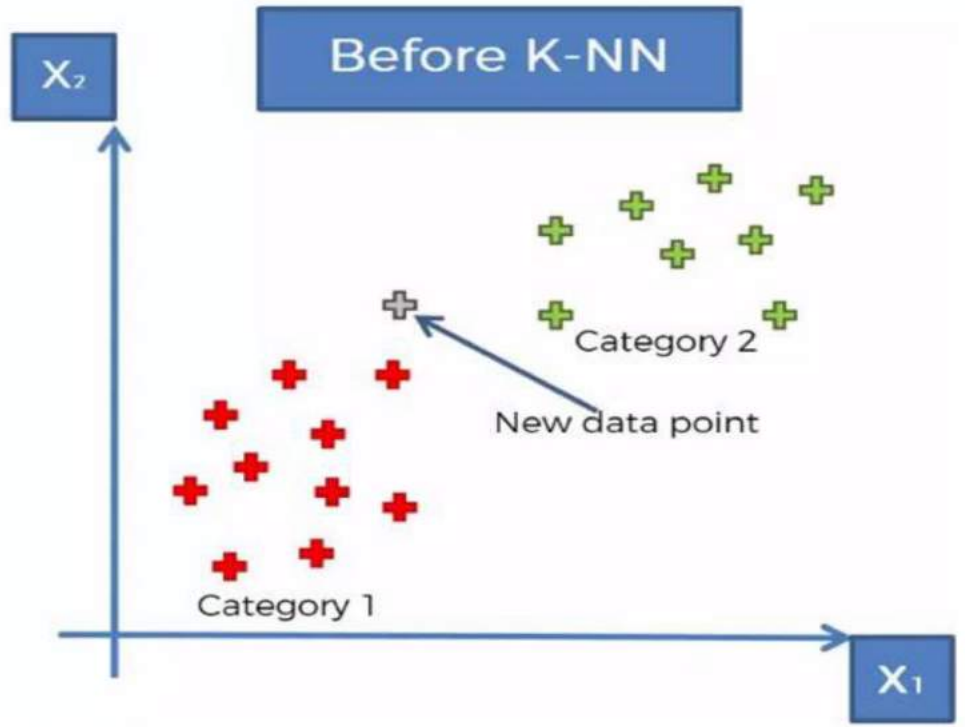
- A powerful classification algorithm used in pattern recognition.
- K nearest neighbors store all available cases and classifies new cases based on a *similarity measure*(e.g **distance function**)
- A non-parametric lazy learning algorithm (An Instance- based Learning method).

KNN: Classification Approach

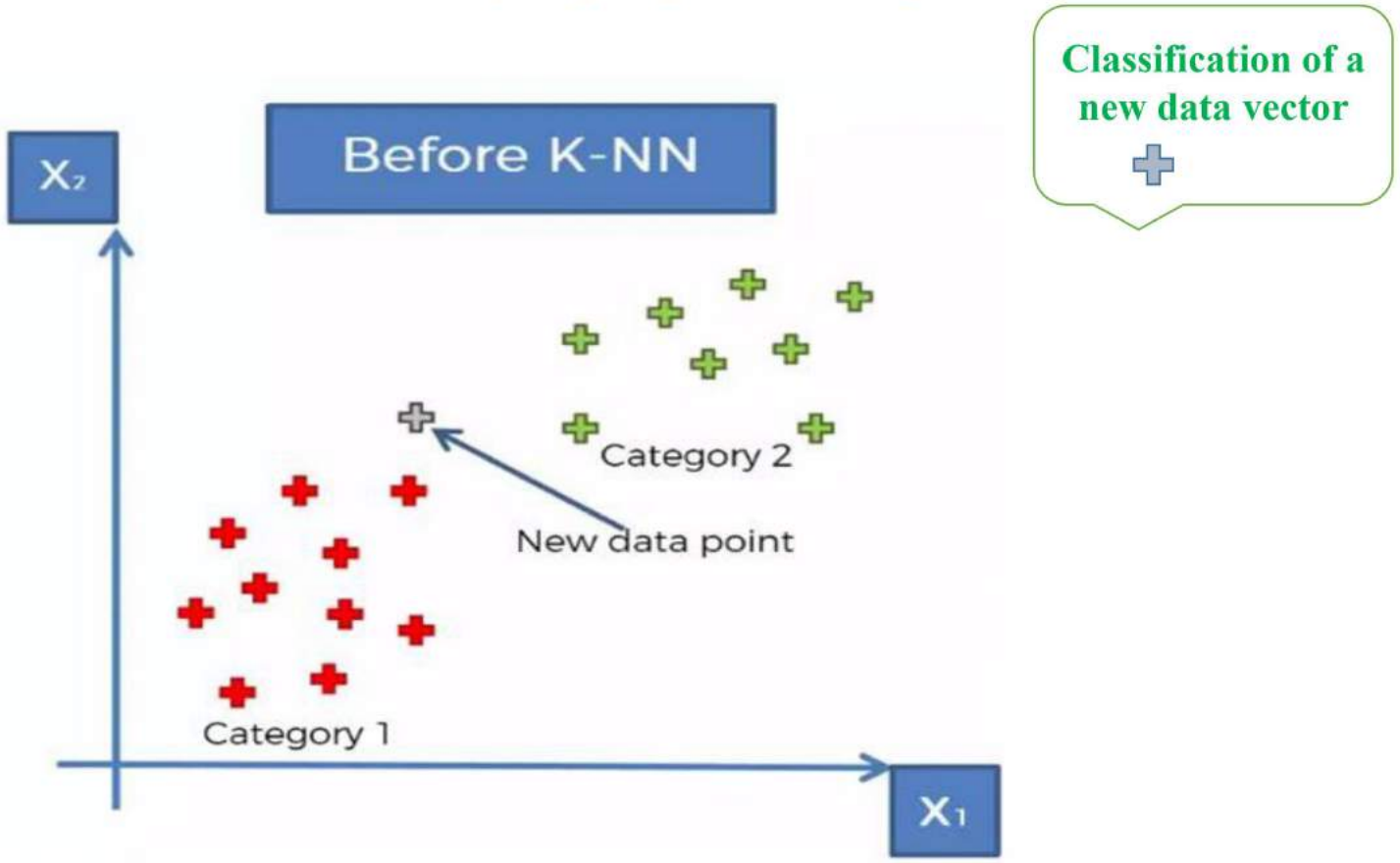
- An object (a new instance) is classified by a majority votes for its neighbor classes.
- The object is assigned to the most common class amongst its K nearest neighbors.(*measured by a distant function*)



Let us understand through graphical representation..!!



Let us understand through graphical representation..!!

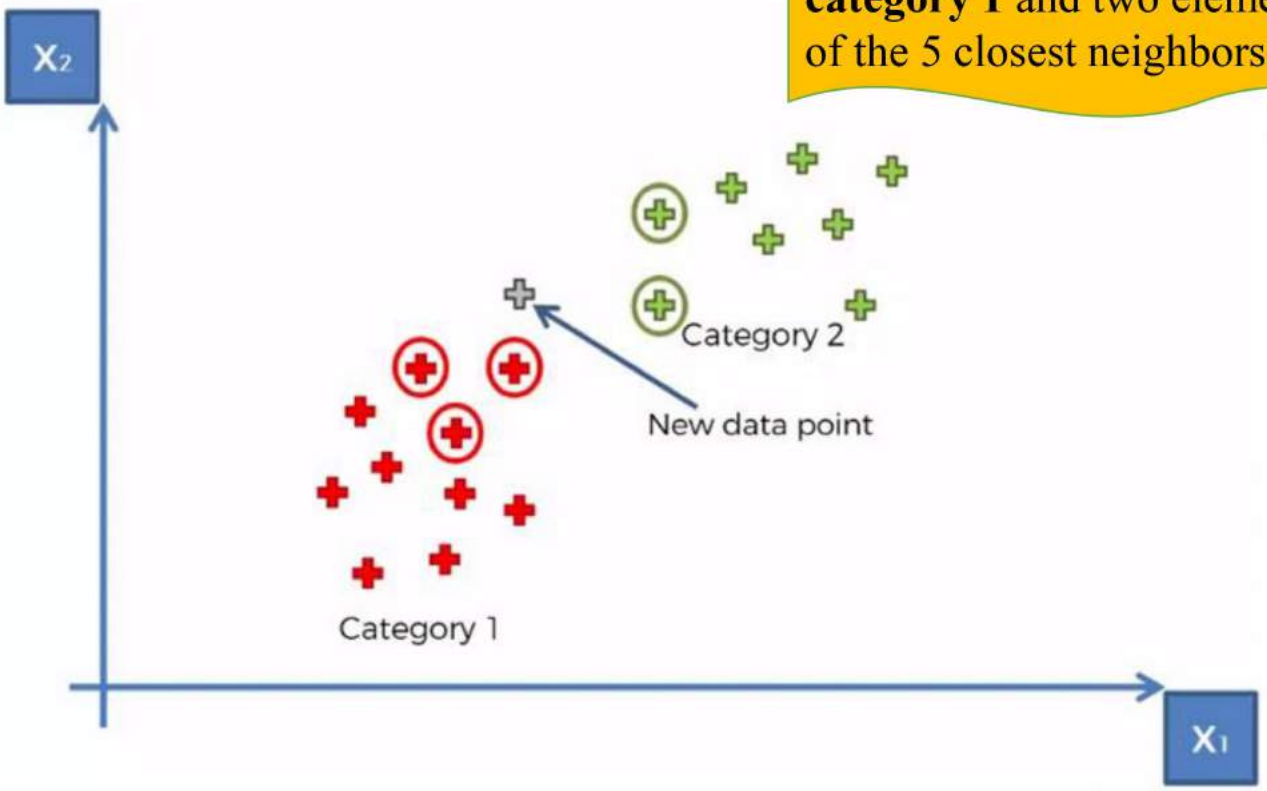


Let us understand through graphical representation..!!

Taking for the example that K = 5, we mark the 5 closest neighbors to the new element

- Selection of the K = 5 nearest neighbors to the new element

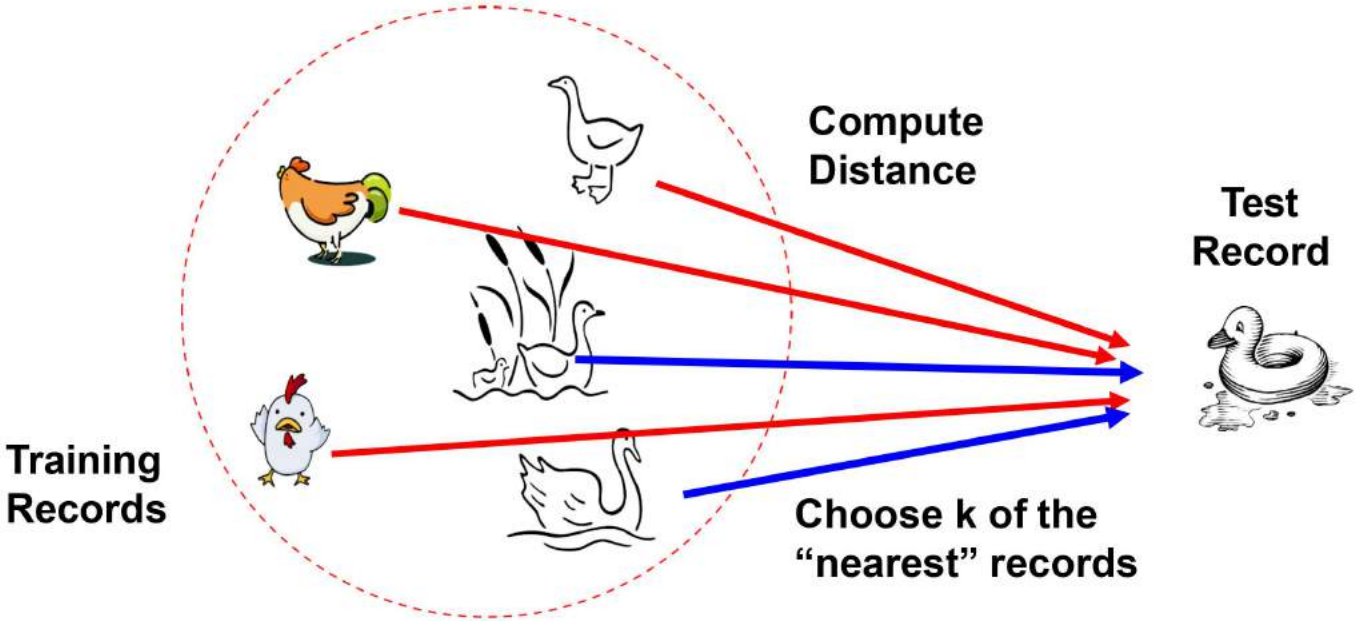
We count that there are 3 elements of **category 1** and two elements of **category 2** of the 5 closest neighbors



- 3 items from category 1 and
- 2 items from category 2

- Therefore, the category with the most elements counted is category 1, so the new element is assigned to the **category 1.**

Distance Measure



Distance measure for continuous variables

Euclidean $\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$

Manhattan $\sum_{i=1}^{k}|x_i - y_i|$

Minkowski $\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$

Distance Between Neighbors

- Calculate the distance between new example (E) and all examples in the training set.

- *Euclidean* distance between two examples.
  - $X = [x_1, x_2, x_3, .., x_n]$
  - $Y = [y_1, y_2, y_3, ..., y_n]$
  - The Euclidean distance between X and Y is defined

$$D(X,Y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Working of KNN Algorithm

- **Step 1** – For implementing any algorithm, we need dataset. So during the first step of KNN, we must load the training as well as test data.

- **Step 2** – Next, we need to choose the value of K i.e. the nearest data points. K can be any integer.

- **Step 3** – For each point in the test data do the following –

- **3.1** – Calculate the distance between test data and each row of training data with the help of any of the method namely: Euclidean, Manhattan or Hamming distance. The most commonly used method to calculate distance is Euclidean.

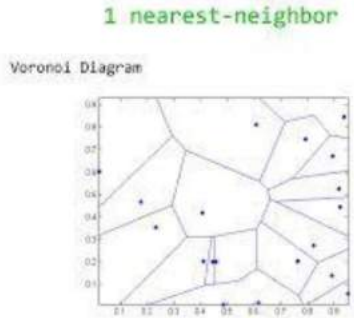
Continued..

- **Step 3** – For each point in the test data do the following –
- **3.1** – Calculate the distance between test data and each row of training data with the help of any of the method namely: Euclidean, Manhattan or Hamming distance. The most commonly used method to calculate distance is Euclidean.
- **3.2** – Now, based on the distance value, sort them in ascending order.
- **3.3** – Next, it will choose the top K rows from the sorted array.
- **3.4** – Now, it will assign a class to the test point based on most frequent class of these rows.
- **Step 4** – End

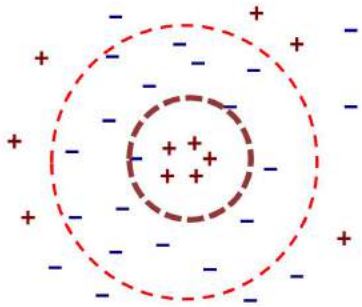
EXAMPLE – 3NN

Customer	Age	Income	No. credit cards	Class	Distance from John
George	35	35K	3	No	sqrt $[(35-37)^2+(35-50)^2+(3-2)^2]=15.16$
Rachel	22	50K	2	Yes	sqrt $[(22-37)^2+(50-50)^2+(2-2)^2]=15$
Steve	63	200K	1	No	sqrt $[(63-37)^2+(200-50)^2+(1-2)^2]=152.23$
Tom	59	170K	1	No	sqrt $[(59-37)^2+(170-50)^2+(1-2)^2]=122$
Anne	25	40K	4	Yes	sqrt $[(25-37)^2+(40-50)^2+(4-2)^2]=15.74$
John	37	50K	2	YES	

How to choose K?



- If K is too small it is sensitive to noise points.
- Larger K works well. But too large K may include majority points from other classes.

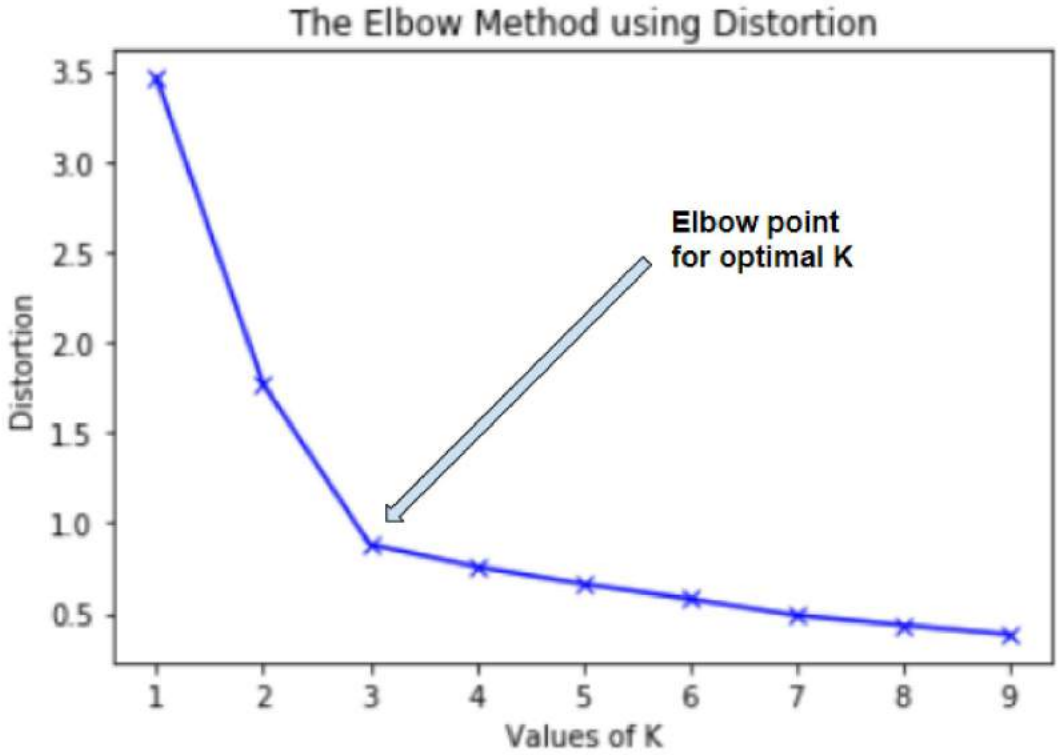


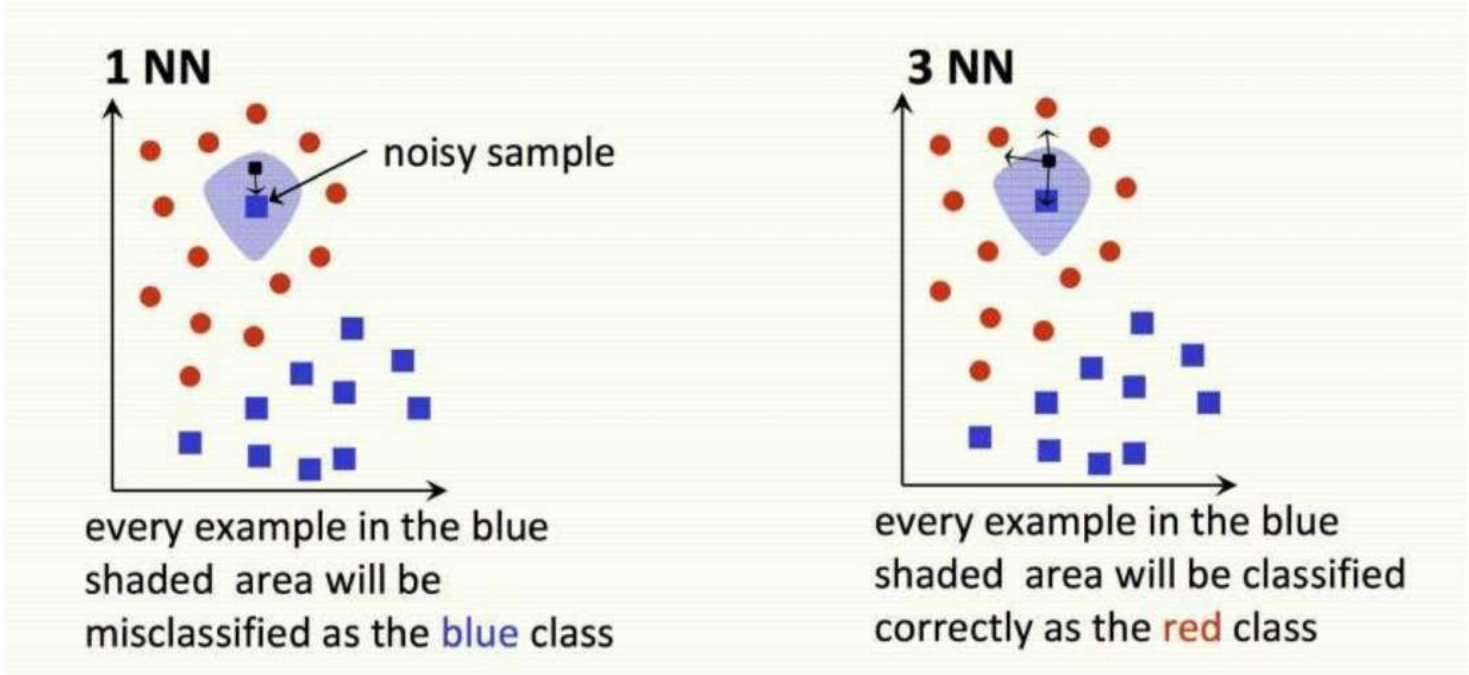
- Rule of thumb is K < sqrt(n), n is number of examples.

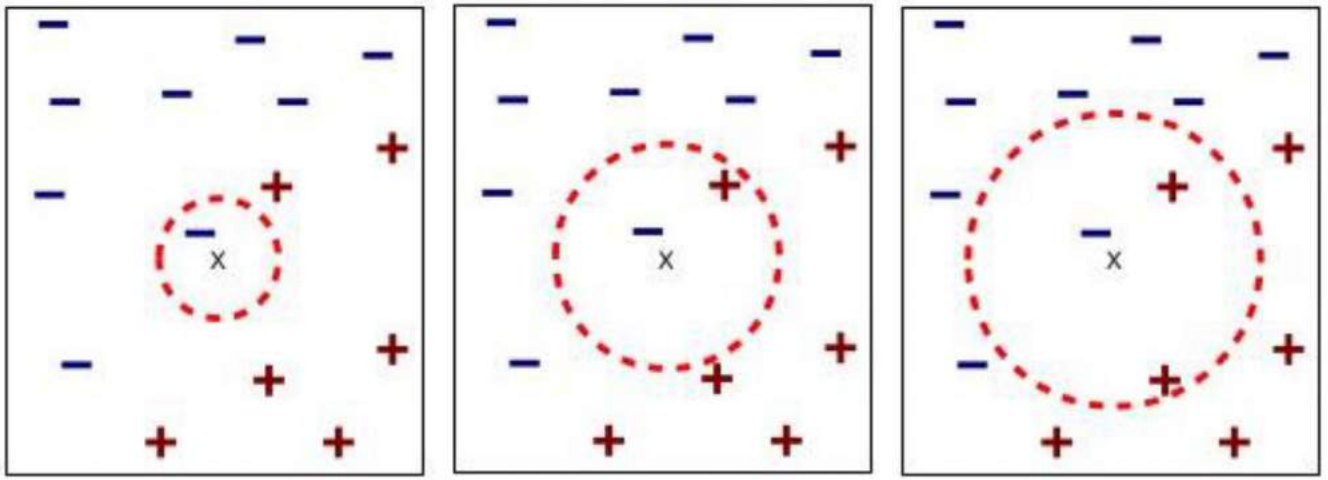
Some more approaches can be:-

- Square Root Method
- Cross Validation (ELBOW METHOD)
- Domain knowledge
- Odd number concept

Elbow Method




1 NN
noisy sample
every example in the blue shaded area will be misclassified as the blue class
3 NN
every example in the blue shaded area will be classified correctly as the red class



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

K-nearest neighbors of a record x are data points that have the k smallest distance to x

Strengths of KNN

- Very simple and intuitive.
- Can be applied to the data from any distribution.
- Good classification if the number of samples is large enough.

Weakness of KNN

- Takes more time to classify a new example.
- Need to calculate and compare distance from new example to all other examples.
- Choosing k may be tricky.
- Need large number of samples for accuracy.

THANK YOU!