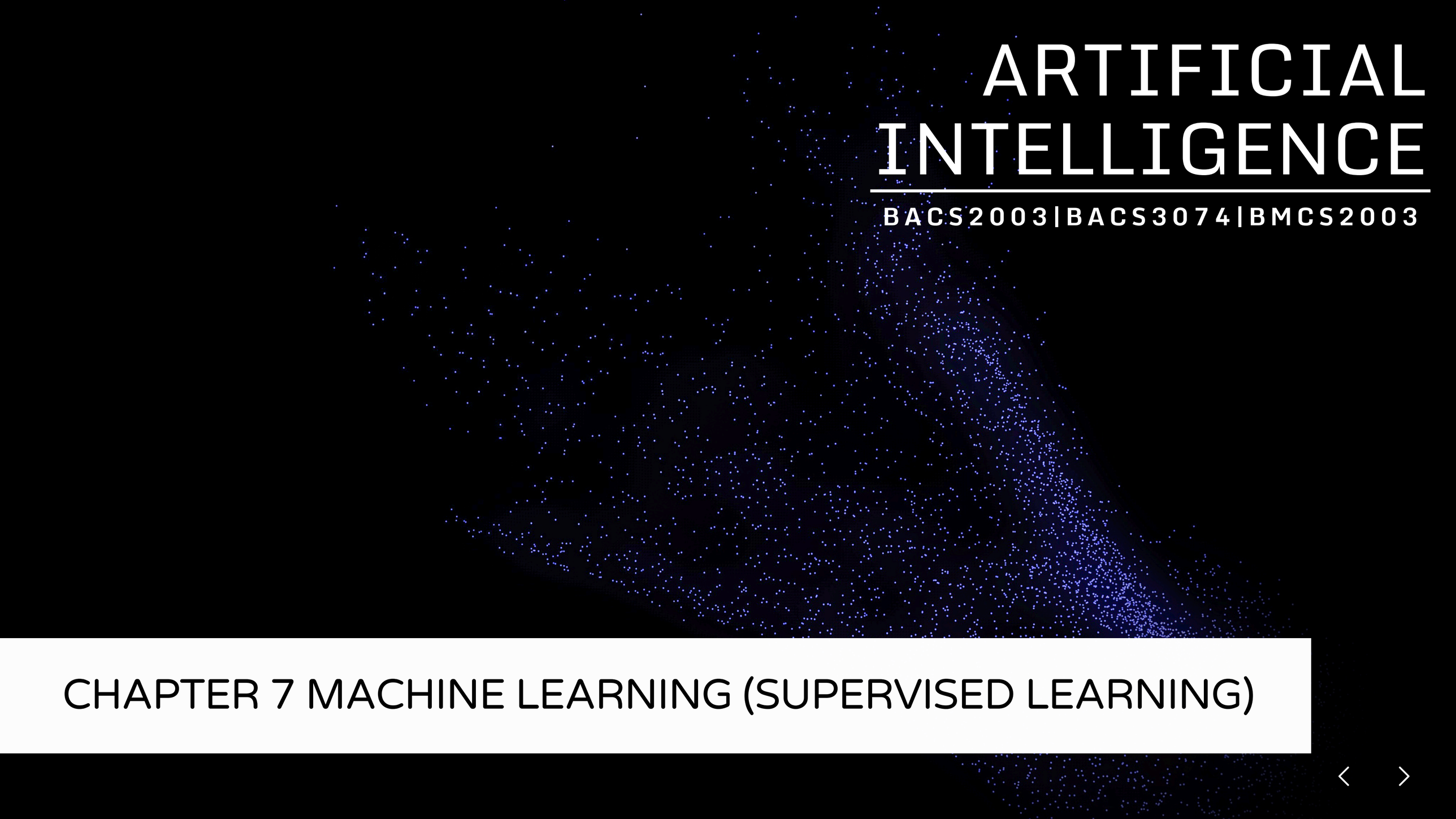
ARTIFICIAL INTELLIGENCE

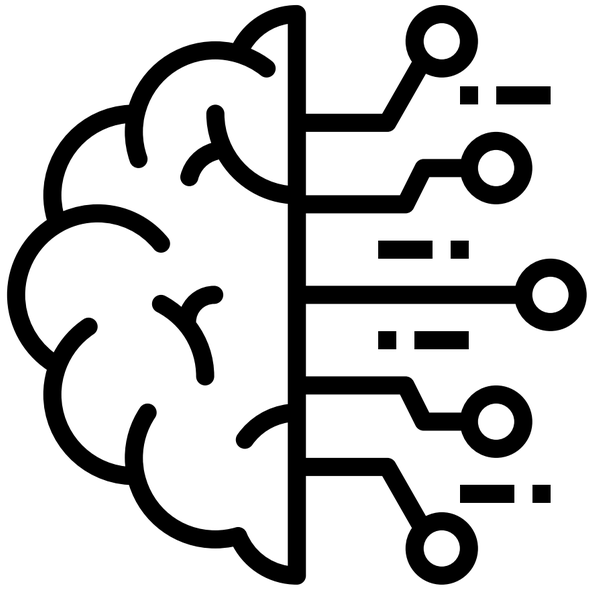
BACS2003|BACS3074|BMCS2003



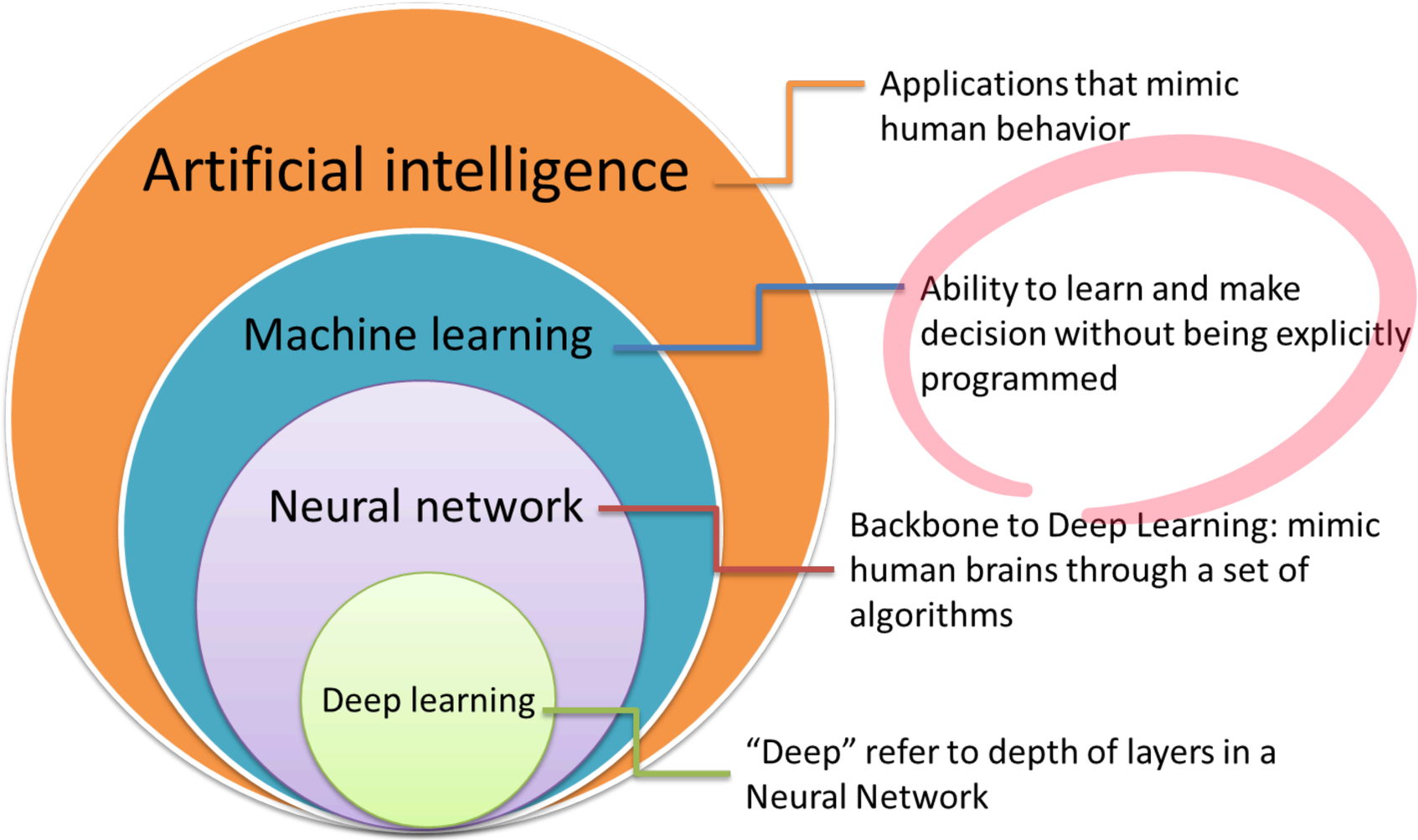
CHAPTER 7 MACHINE LEARNING (SUPERVISED LEARNING)

OUTCOMES

1. **Relationships of AI, ML, DL & Types of Machine Learning (ML)**
2. **Process flow of Supervised ML (SML)**
3. **Features & Labels**
4. **Overfitting**
5. **SML Algorithms**
6. **Assessing Classification Performance**



TERMINOLOGY



TYPE OF MACHINE LEARNING

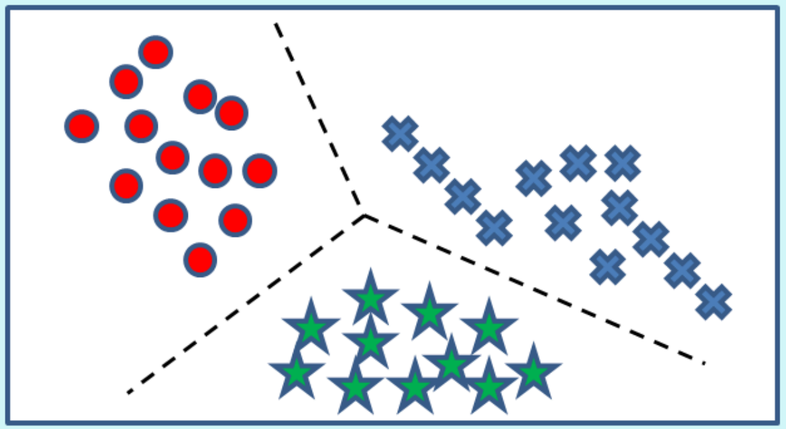
- Usually divided into two main types:
  - Supervised
  - Unsupervised

Will be covered

- Uncommon types:
  - Semi-supervised
  - Reinforcement learning

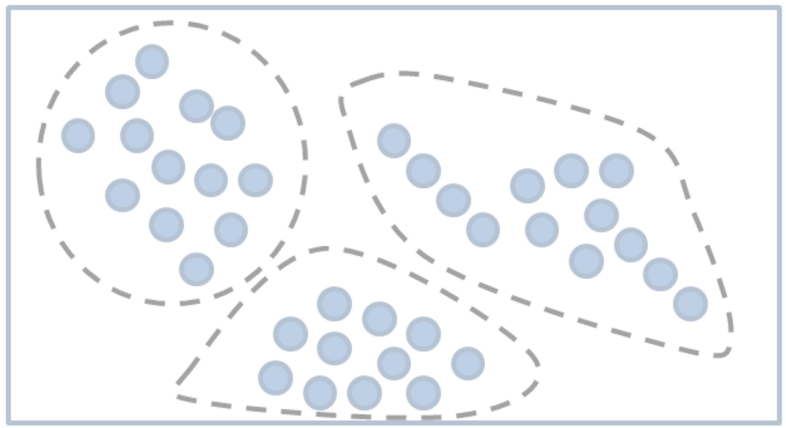
Extra reading and searching

TYPE OF MACHINE LEARNING



Supervised learning

Today



Unsupervised learning

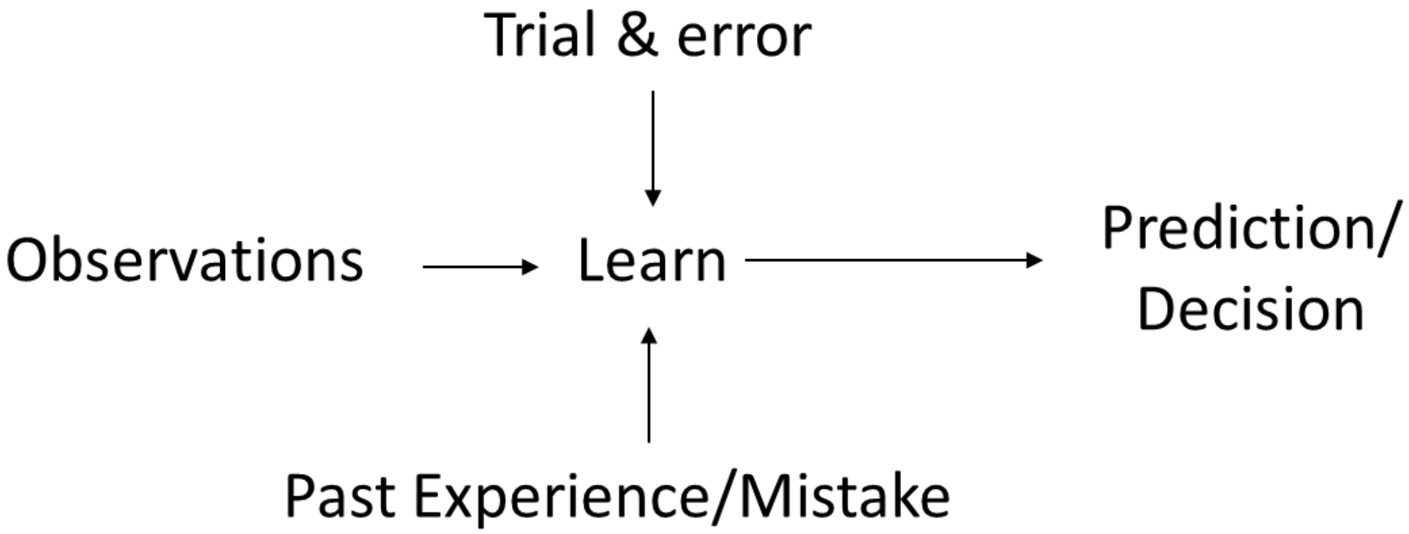
Next week

MACHINE LEARNING: DEFINITION

Machine Learning is a set of methods that can automatically detect patterns in data, and then use the uncovered patterns to predict future data, or to perform other kinds of decision making under uncertainty.

Kevin P. Murphy

HOW DOES HUMAN LEARN?

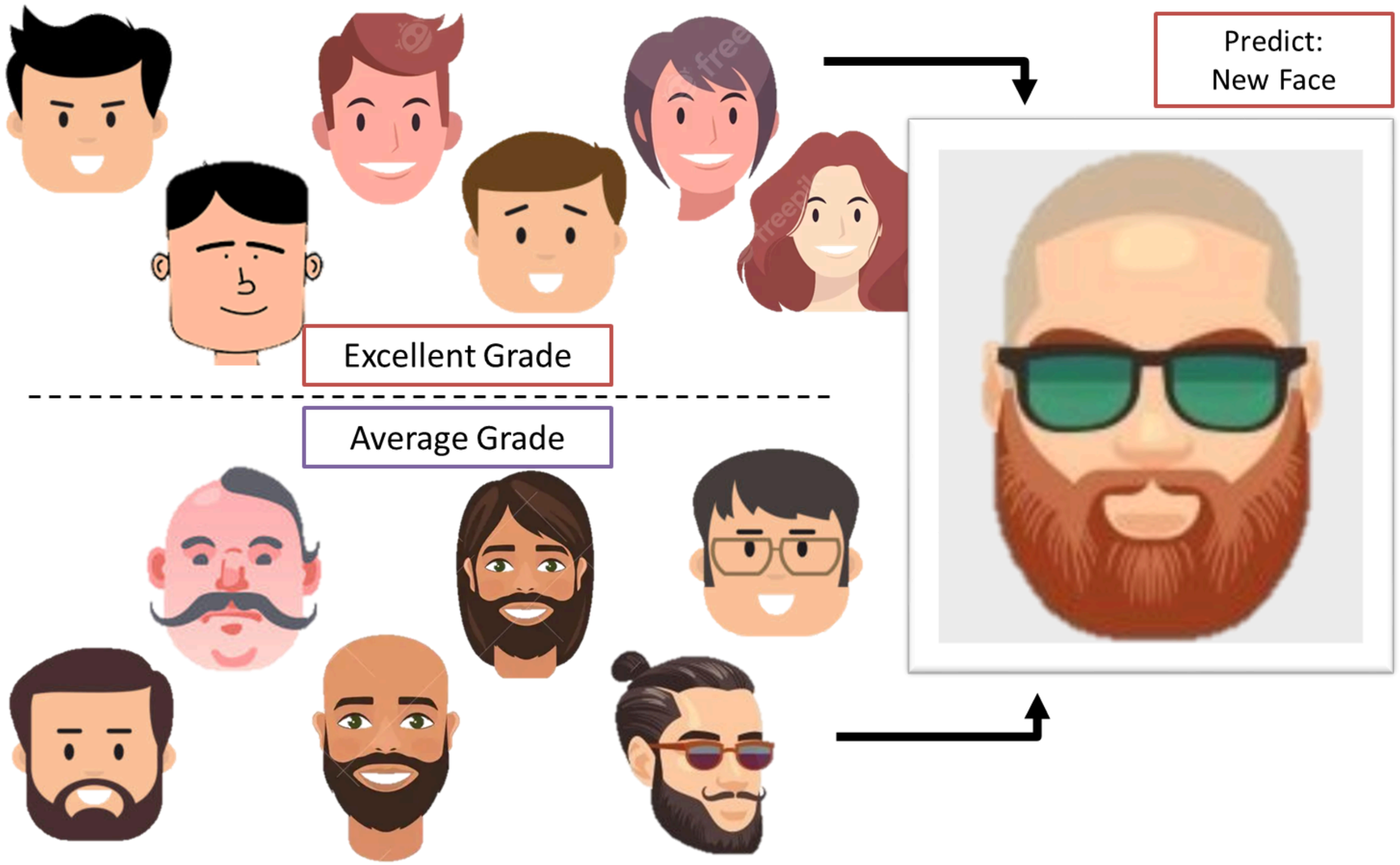


HOW HUMAN LEARN?

Excellent Grade

Average Grade

HOW HUMAN LEARN?



HOW HUMAN LEARN?

Excellent Grade

Average Grade

- Hair length
- Hair style
- Hair color
- Bespectacled
- Beard/mustache
- Head size
- Gender
- Skin condition
- Eyes size
- Eyes color
- Teeth alignment
- etc

- Hair length : Short
- Hair style : Neat
- Bespectacled : False
- Beard/mustach : False
- Head size : Rectangle/Sharp
- Gender : Male/Female

- Hair length: Long/False
- Hair style : Fashion
- Bespectacled : True
- Beard/mustache : True
- Head size : Rectangle/Round
- Gender : Male

HOW HUMAN LEARN?

Labels

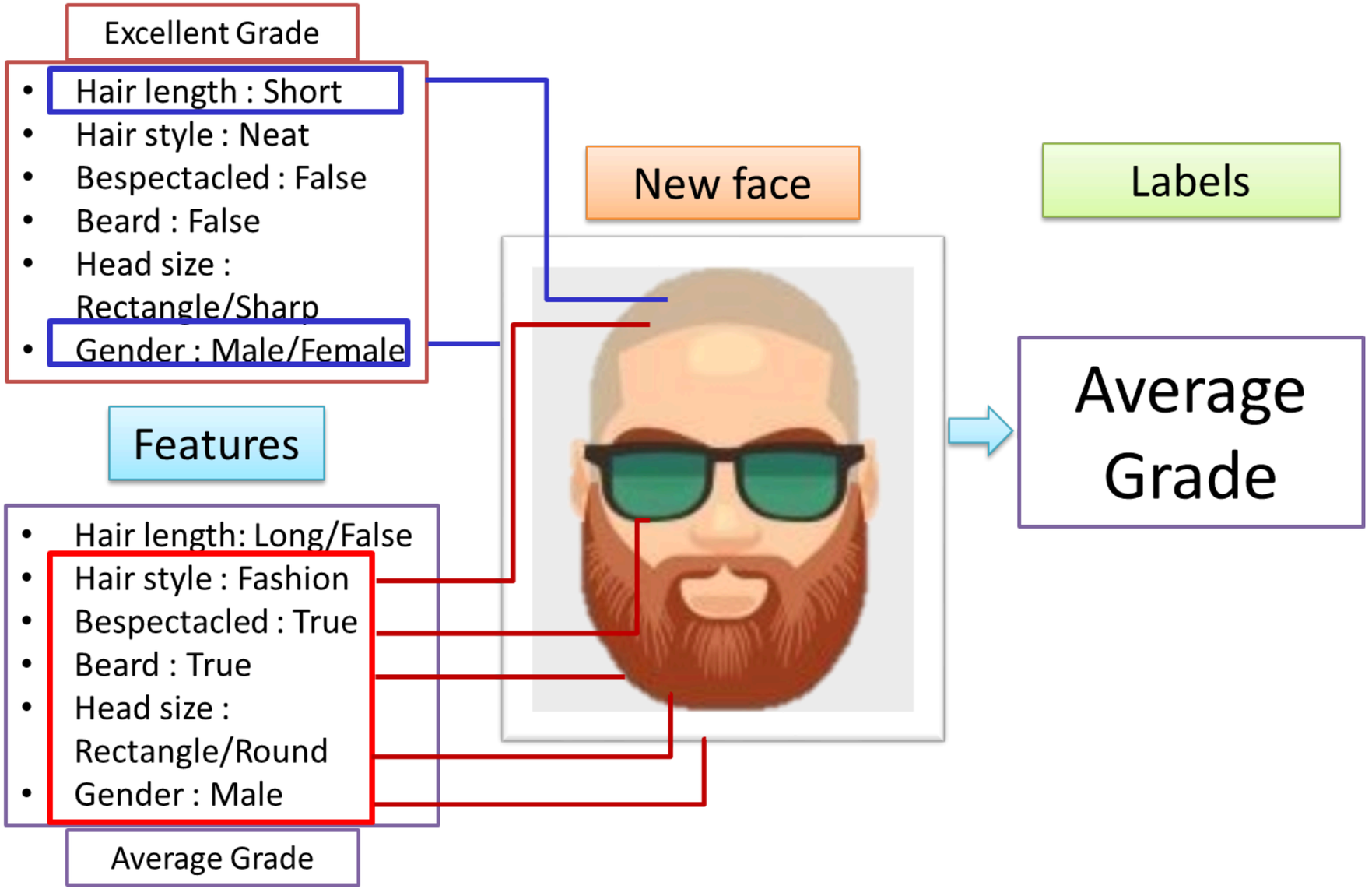
- Hair length
- Hair style
- Hair color
- Bespectacled
- Beard/mus
- Head size
- Gender
- Skin condition
- Eyes size
- Eyes color
- Teeth alignment
- etc

Features

- Hair length : Short
- Hair style : Neat
- Bespectacled : False
- Beard/mustach : False
- Head size : Rectangle/Sharp
- e/Female

- Hair length: Long/False
- Hair style : Fashion
- Bespectacled : True
- Beard/mustache : True
- Head size : Rectangle/Round
- Gender : Male

HOW HUMAN LEARN?



SML : FEATURES AND LABELS

Machine doesn’t know what fruits are these



x (Features) | *y* (Label)

Color	Size	Fruit
Red	Big	Apple
Orange	Big	Orange
Red	Small	Grapes
Red	Big	Apple
Orange	Big	Orange

IN GENERAL

- Classification/prediction is like human learn from past experiences.

- Computer does not has “experiences” so it learns from data, which represent some “past experiences” of an application domain.

SUPERVISED LEARNING

- The data are labelled with pre-defined classes. It is like that a “teacher” gives the classes (supervision).
- Goal: learn a mapping from inputs x to outputs y (relationship), given a labeled set of input-output pairs. $\mathscr{D}$ is the training set and n is the number of training examples.

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$$

EXAMPLE: DATA (LOAN APPLICATION)

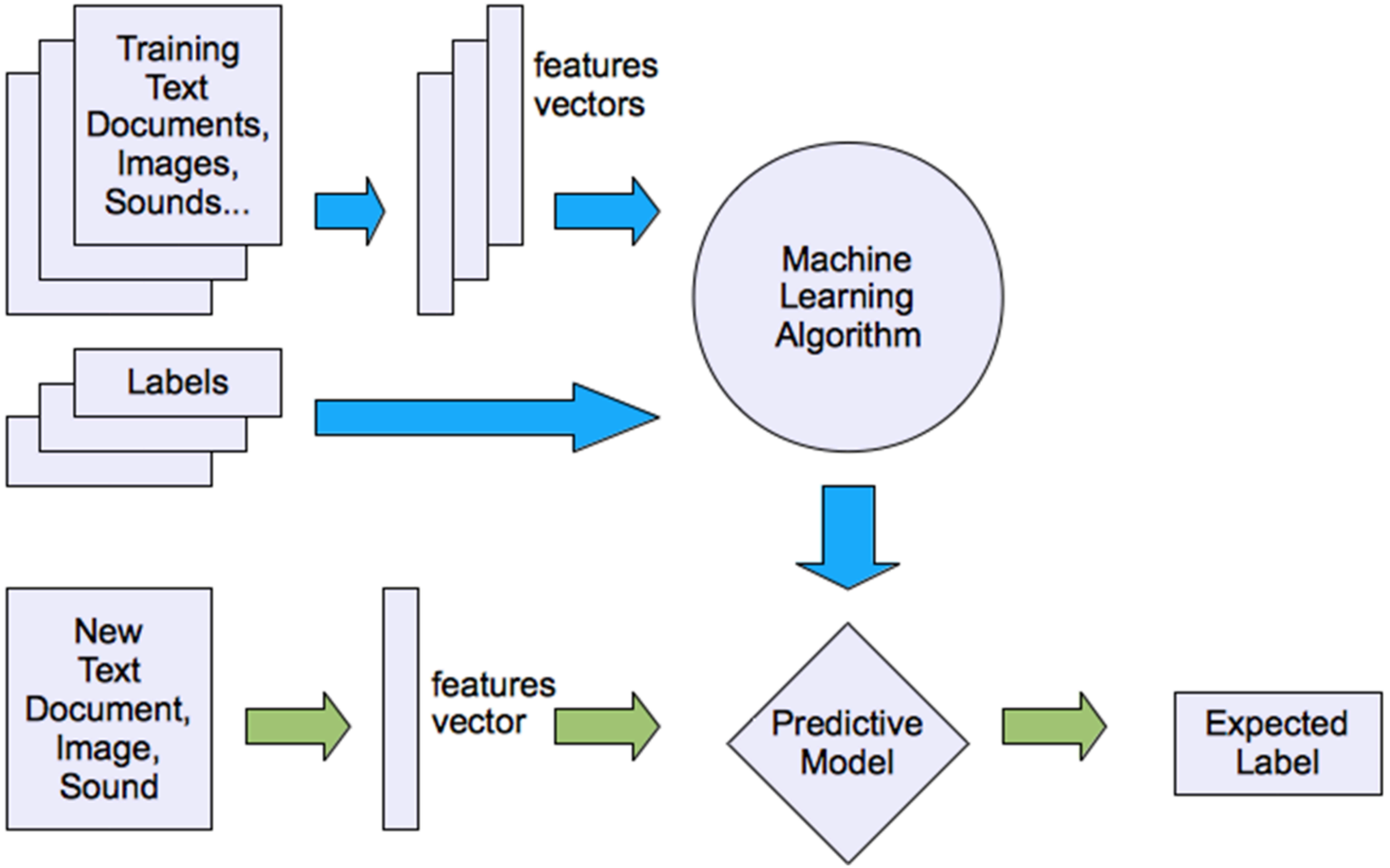
Feature = A1,A2,A3,A4

ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	No
2	young	false	false	good	No
3	young	true	false	good	Yes
4	young	true	true	fair	Yes
5	young	false	false	fair	No
6	middle	false	false	fair	No
7	middle	false	false	good	No
8	middle	true	true	good	Yes
9	middle	false	true	excellent	Yes
10	middle	false	true	excellent	Yes
11	old	false	true	excellent	Yes
12	old	false	true	good	Yes
13	old	true	false	good	Yes
14	old	true	false	excellent	Yes
15	old	false	false	fair	No

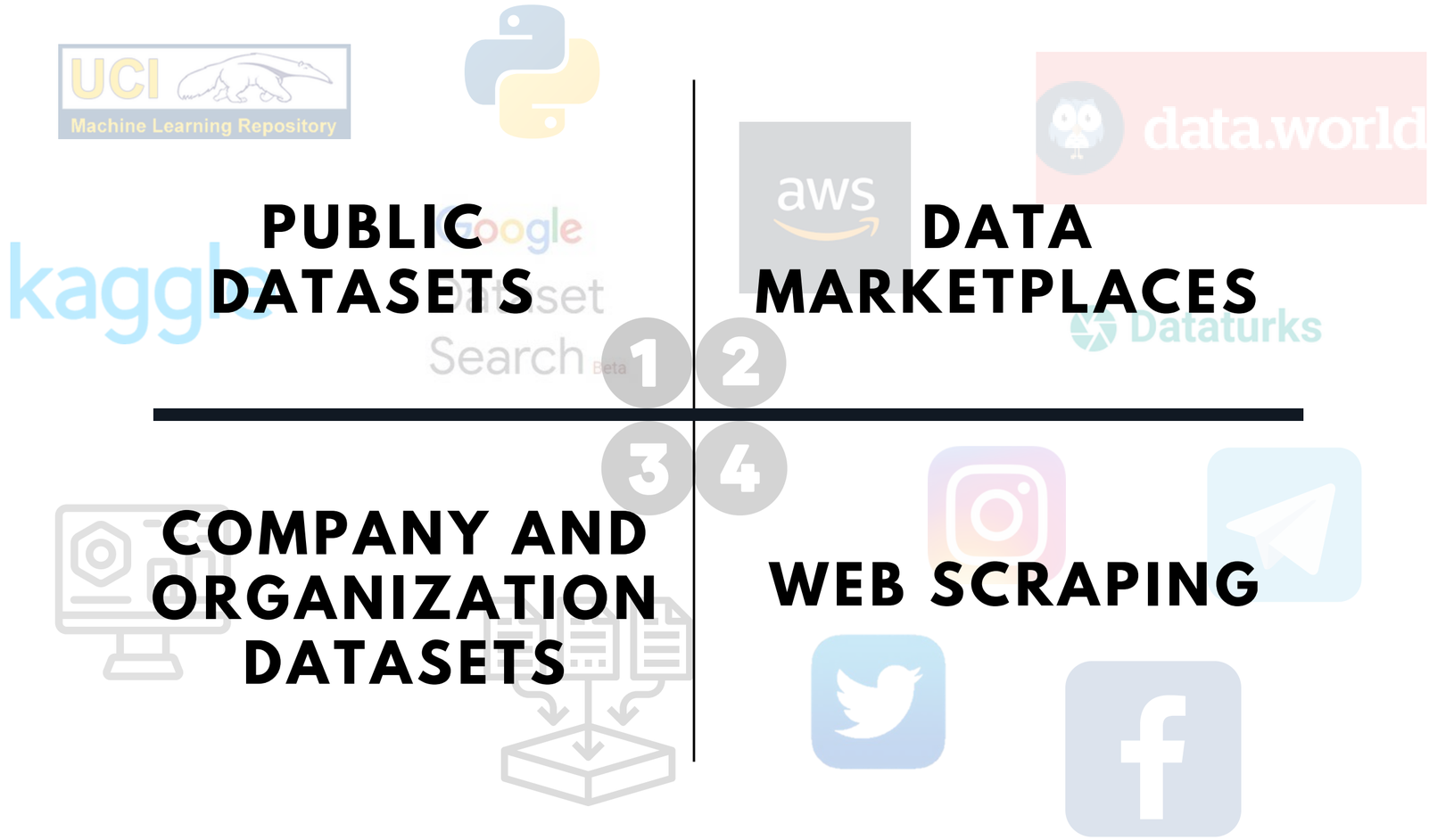
$$\mathcal{D}_i = \{A_1, A_2, A_3, A_4, y_i\}$$

$$\mathcal{D}_1 = \{Age = Middle, Has\ Job = False, Own\ House = False, Credit\ Rating = fair, Class = No\}$$

GENERAL PROCESS FLOW OF SUPERVISED LEARNING



WHERE TO RETRIEVE DATASET?



PREPROCESSING

Data Cleaning

Handling missing values by by either removing the corresponding samples or filling in the missing values with techniques such as mean, median, or mode imputation. Address outliers by using Winsorization technique or outlier imputation

Data Integration

If you have multiple data sources or datasets, you may need to integrate them into a single dataset. This typically involves handling inconsistencies in attribute names, resolving conflicts in data formats, and merging the data based on common identifiers.

Data Transform-ation

Common transformations include scaling numerical features to a similar range (e.g., using normalization or standardization), encoding categorical variables into numerical representations (e.g., one-hot encoding), and transforming skewed distributions

Imbalanc-ed Data

Technique such as oversampling the minority class, undersampling the majority class, or using advanced algorithms like SMOTE (Synthetic Minority Over-sampling Technique) can be employed to address the imbalance.

Time-Series Data

Handling missing or irregular timestamps, resampling or interpolating the data to a regular time interval, and creating lag features or rolling windows for capturing temporal patterns.

MACHINE LEARNING ALGORITHMS

CATEGORICAL / NOMINAL (DISCRETE LABELS)

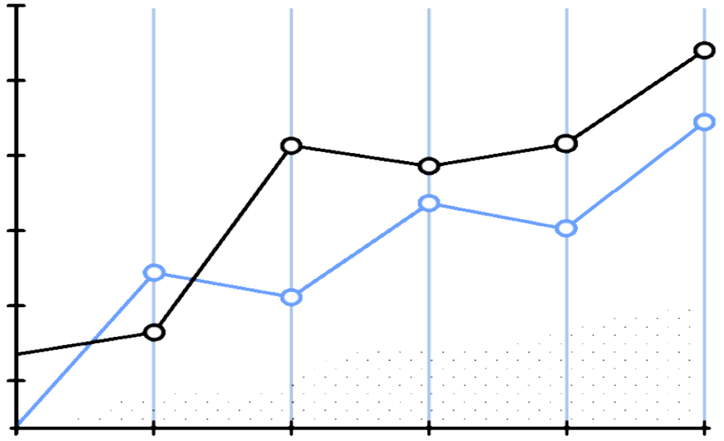
Classification

LOGISTIC REGRESSION
DECISION TREES (CLASSIFICATION)
RANDOM FOREST
SUPPORT VECTOR MACHINES
NAIVE BAYES
K-NEAREST NEIGHBOURS

CONTINUOUS VALUES

Regression

LINEAR REGRESSION
GRADIENT BOOSTING
ADABOOST
DECISION TREE (REGRESSION)



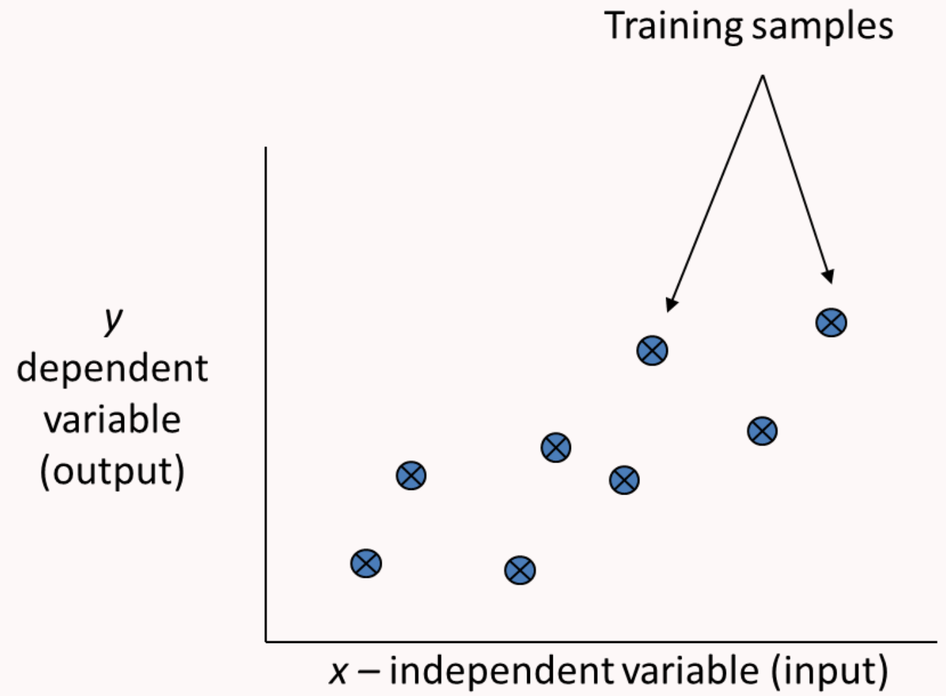
REAL WORLD APPLICATION

- Document classification
- Email spam filtering
- Object classification
- Face detection and recognition
- Fraud detection (credit card, insurance)
- Disease prediction and classification
- Stock price prediction

LINEAR REGRESSION

In regression the output is continuous real value

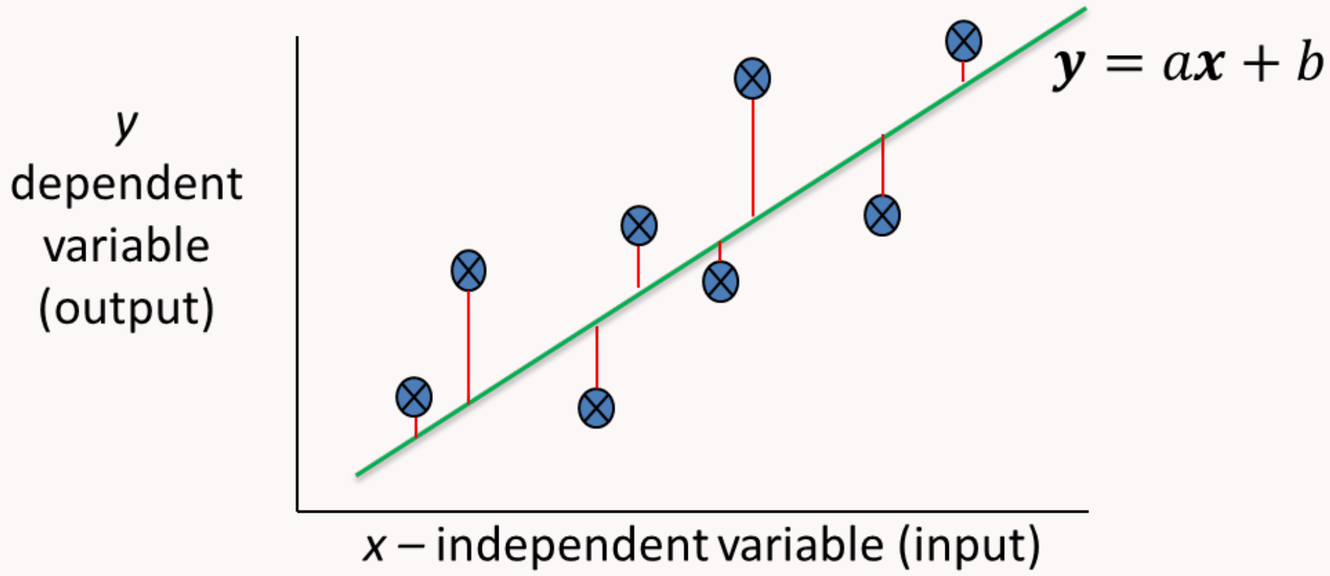
- Function Approximation



LINEAR REGRESSION

Many models could be used – Simplest is linear regression

- Fit data with the best hyper-plane which "goes through" the points
- For each point the differences between the predicted point and the actual observation is the residue



LINEAR REGRESSION

	Player payroll, x (in $1,000,000s)	Mean attendance, y (in thousands)
Anaheim	46.6	24.69
Baltimore	73.4	38.15
Boston	109.6	32.47
Chicago White Sox	62.4	21.85
Cleveland	92.0	39.26
Detroit	49.8	23.70
Kansas City	35.6	19.01
Minnesota	24.4	21.98
New York Yankees	109.8	40.25
Oakland	33.8	26.54
Seattle	75.7	18.33
Tampa Bay	55.0	16.05
Texas	88.5	34.94
Toronto	75.8	23.70

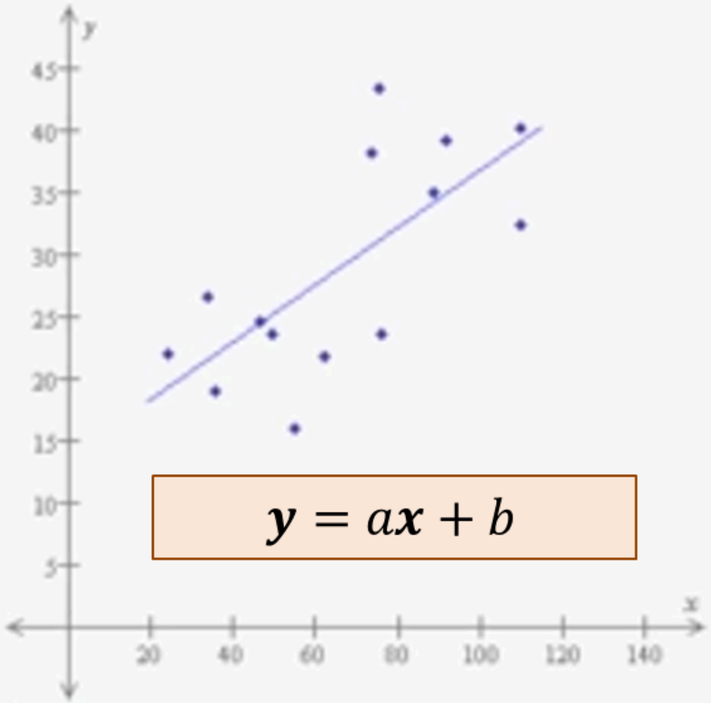


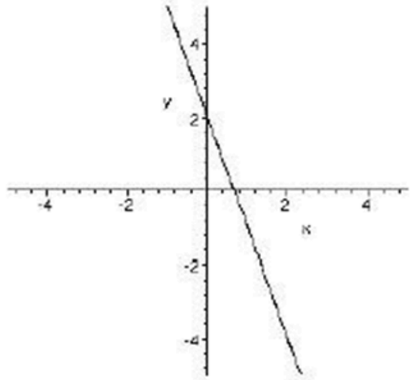
Figure 1

$$when\ x = 33.8$$

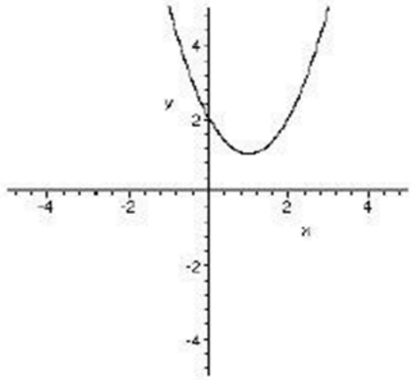
$$y = 0.23x + 13.82$$

$$y = 21.6$$

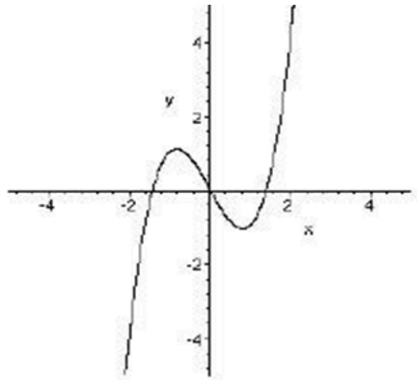
POLYNOMIAL REGRESSION & NON-LINEAR REGRESSION



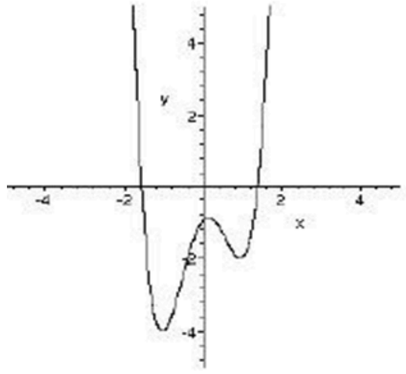
$f(x) = -3x + 2$



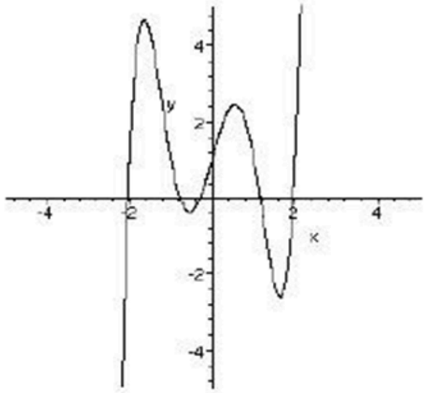
$g(x) = x^2 - 2x + 2$



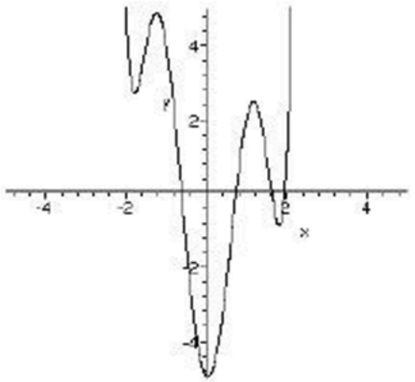
$h(x) = x^3 - 2x$



$F(x) = 2x^4 - 4x^2 + x - 1$



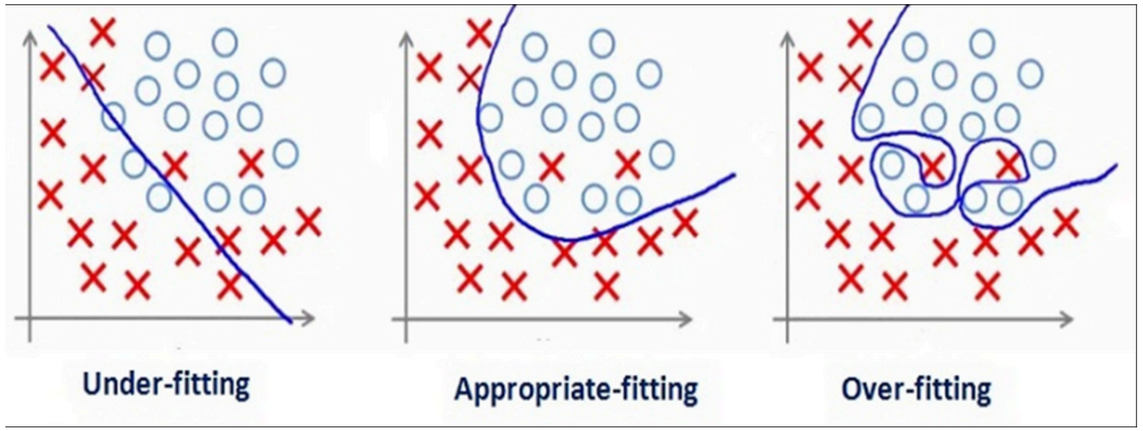
$G(x) = x^5 - 5x^3 + 4x + 1$

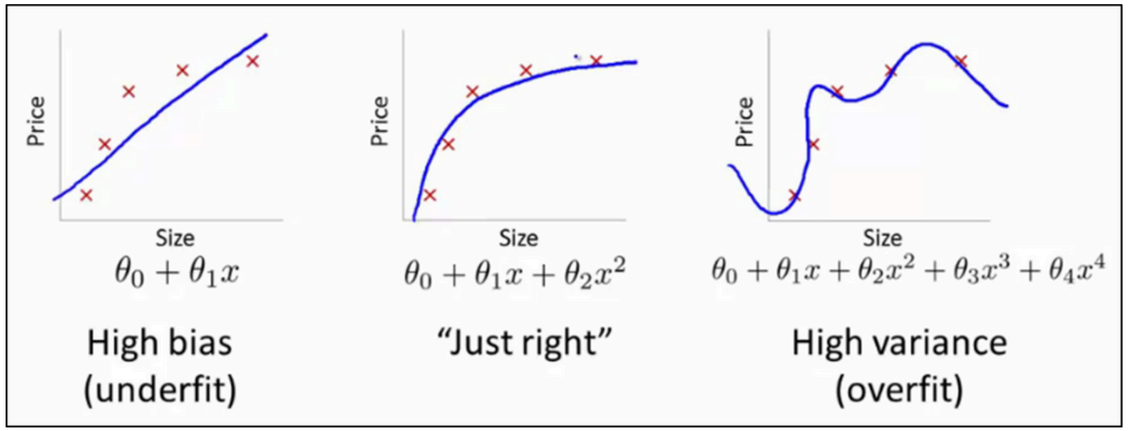


$H(x) = x^6 - 7x^4 + 14x^2 - x - 5$

LINEAR REGRESSION : OVERFITTING

- Number of parameters estimated from the data must be considerably less than the number of data points
- Advisable to choose the degree of polynomial as low as possible – often a simple linear relationship is assumed





CLASSIFICATION METHODS

1. K Nearest Neighbour
2. Decision Tree
3. Support Vector Machine
4. Bayesian Classification

K-NEAREST NEIGHBOUR

DISTANCE-BASED CLASSIFIER

- Simple yet effective algorithm that makes predictions based on the similarity of data points in a feature space
- The "K" in KNN refers to the number of nearest neighbors considered when making a prediction
- KNN can deal with complex and arbitrary decision boundaries.
- Despite its simplicity, researchers have shown that the classification accuracy of KNN can be quite strong and in many cases as accurate as those elaborated methods.
- KNN is slow at the classification time
- KNN does not produce an understandable model

K-NEAREST NEIGHBOUR

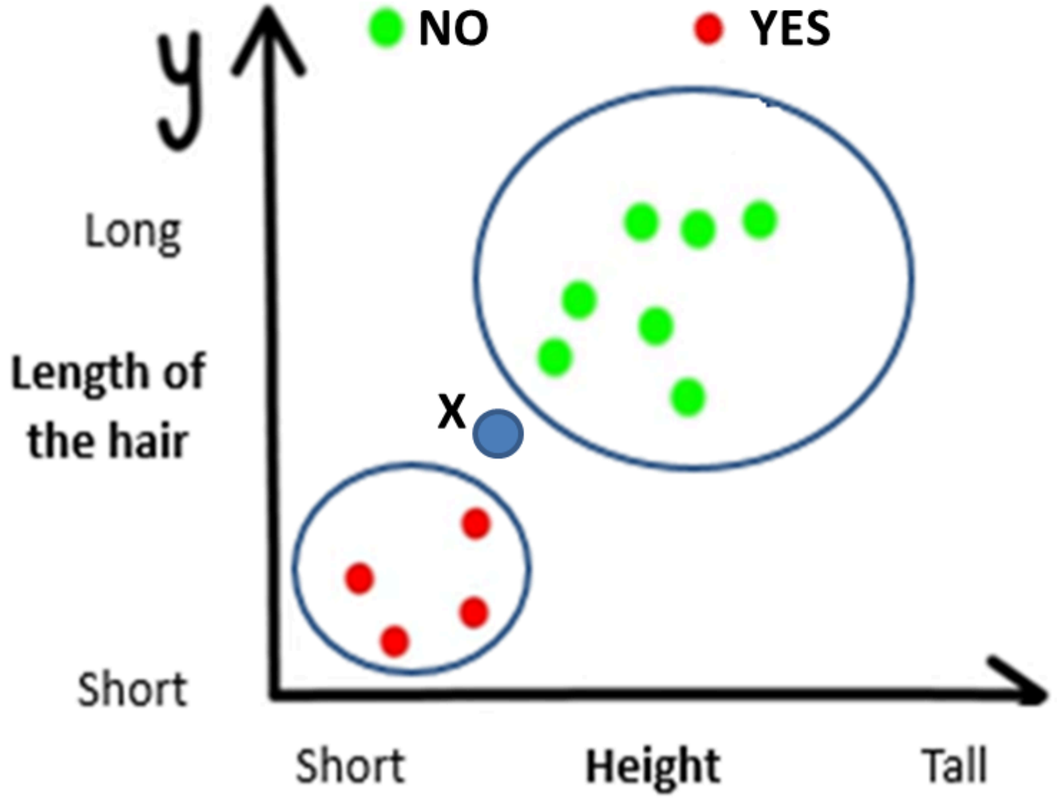
Algorithm:

1. Compute the distance between d and every training sample in $\mathscr{D}$;
2. Choose the k sample in $\mathscr{D}$ that are nearest to d
3. Assign d the class that is the most frequent class in the neighbourhood (or the majority class)

- k is usually chosen empirically via a validation set or cross-validation by trying a range of k values.
- Distance function is crucial, but depends on applications.

K-NEAREST NEIGHBOUR

Cinderella, x



K-NEAREST NEIGHBOUR

Step 1: Measure distance between unseen data, x with dataset

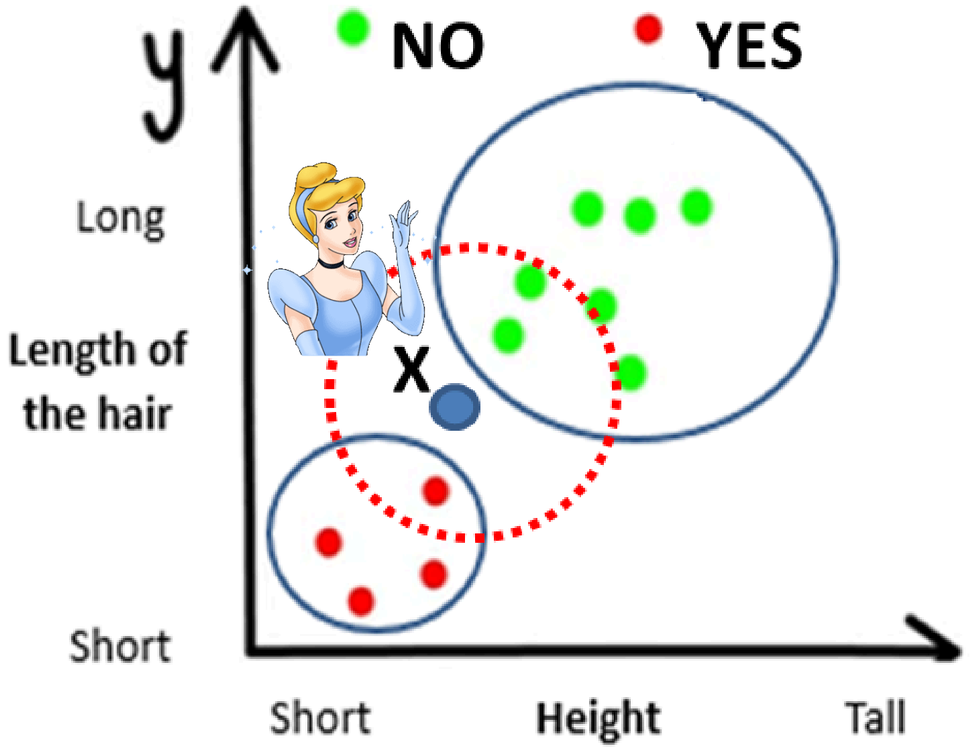
Step 2: Identify “K” value (predefined by user), in this case let say k=3

Step 3: Find the probability that the unseen data, x belongs to

$$Pr(No|\ Cinderella) = 2/3 = 0.67$$
$$Pr(Yes|\ Cinderella) = 1/3 = 0.33$$

Conclusion: Cinderella will reject prince

```
pip install scikit-learn
from sklearn.neighbors import KNeighborsClassifier
...
...
k = 3
knn = KNeighborsClassifier(n_neighbors=k)
knn.fit(X_train, y_train)
predictions = knn.predict(X_test)
```



DECISION TREE

Decision tree

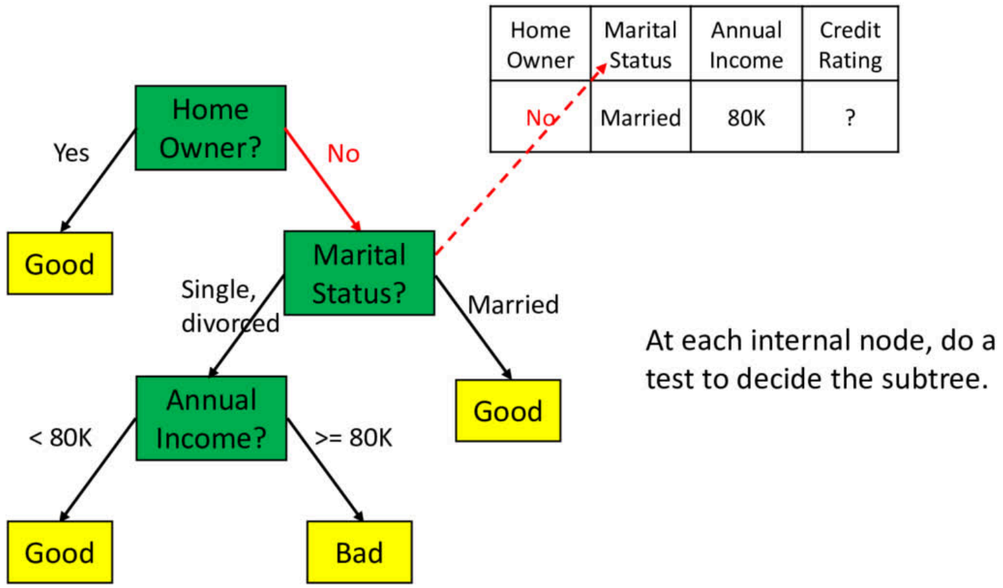
- A flow-chart-like tree structure
- Internal node denotes a test on an attribute
- Branch represents an outcome of the test
- Leaf nodes represent class labels or class distribution

Decision tree generation consists of two phases

- Tree construction
  - At start, all the training examples are at the root
  - Partition examples recursively based on selected attributes
- Tree pruning
  - Identify and remove branches that reflect noise or outliers

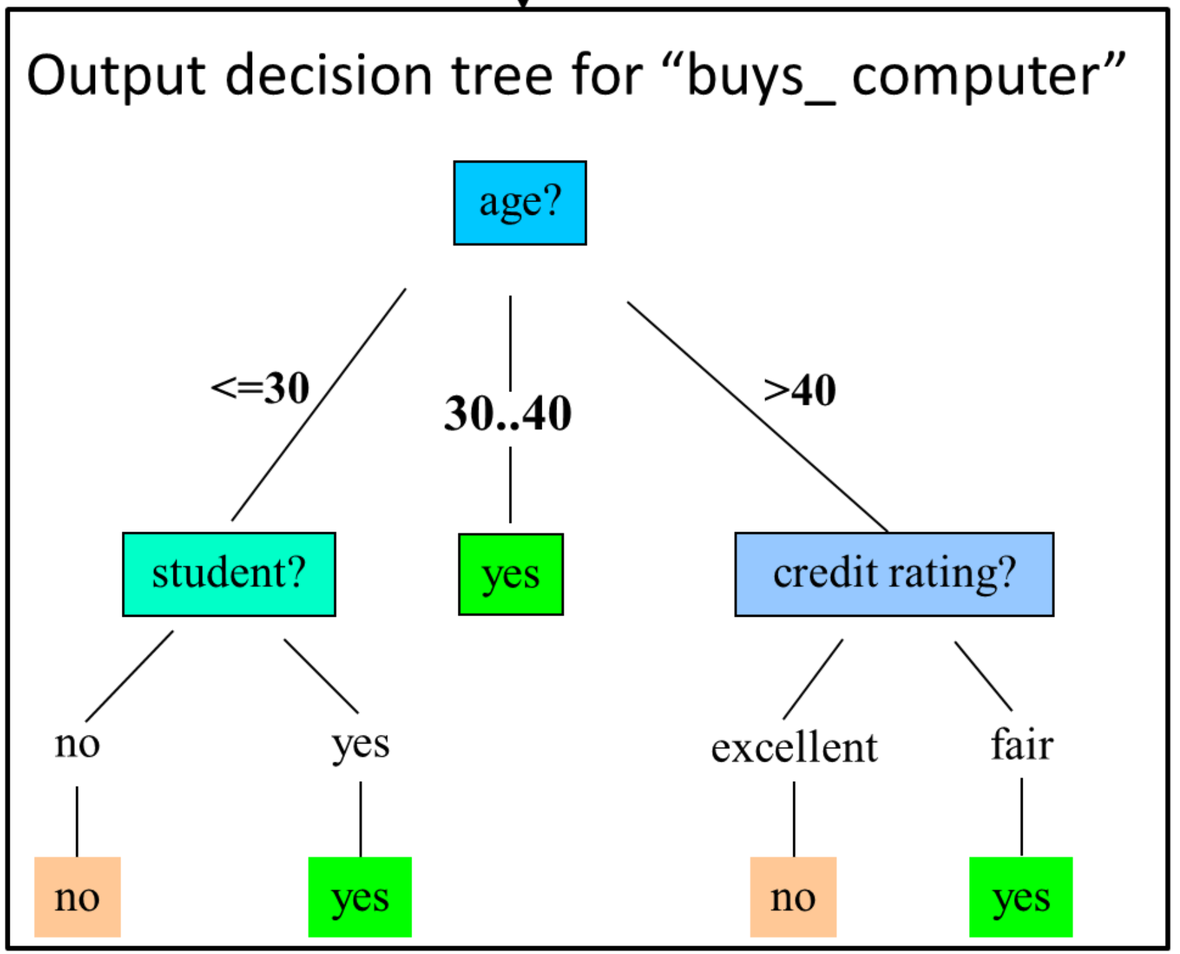
Use of decision tree: Classifying an unknown sample

- Test the attribute values of the sample against the decision tree



DECISION TREE

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no



age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

On the basis of tree constructed in the manner described, classify a test sample:

(age , student, creditrating, buys_computer)
(<=30 , yes , excellent , ?)

Will this student buy computer?

DECISION TREE

- Basic algorithm (a greedy algorithm)
  - Tree is constructed in a **top-down recursive divide-and-conquer manner**
  - At start, all the training examples are at the root
  - Attributes are categorical (**if continuous-valued, they are discretized in advance**)
  - Examples are partitioned recursively based on selected attributes
  - **Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain**)
- Conditions for stopping partitioning
  - All samples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
  - There are no samples left

INFORMATION GAIN CALCULATION (ID3/C4.5)

- Select the attribute with the highest information gain
- Assume there are two classes, ***P*** and ***N*** (yes and no from example)
- Let the set of examples $\mathcal{D}$ contain *p* elements of class ***P*** and *n* elements of class ***N***
  - The **amount of information**, needed to decide if an arbitrary example in *S* belongs to ***P*** or ***N*** is defined as

$$I(p,n) = -\left(\frac{p}{p+n}\log_2\frac{p}{p+n}\right) - \left(\frac{n}{p+n}\log_2\frac{n}{p+n}\right)$$

INFORMATION GAIN CALCULATION (ID3/C4.5)

- Assume that using attribute A, a set $\mathcal{D}$ will be partitioned into sets $\{S_1, S_2, \ldots, S_v\}$
  - If S_i contains p_i examples of *P* and n_i examples of *N*, the **entropy**, or the expected information needed to classify objects in all subtrees S_i is

$$E(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p + n} I(p_i, n_i)$$

- The encoding information that would be gained by branching on *A*

$$Gain(A) = I(p, n) - E(A)$$

ENTROPY

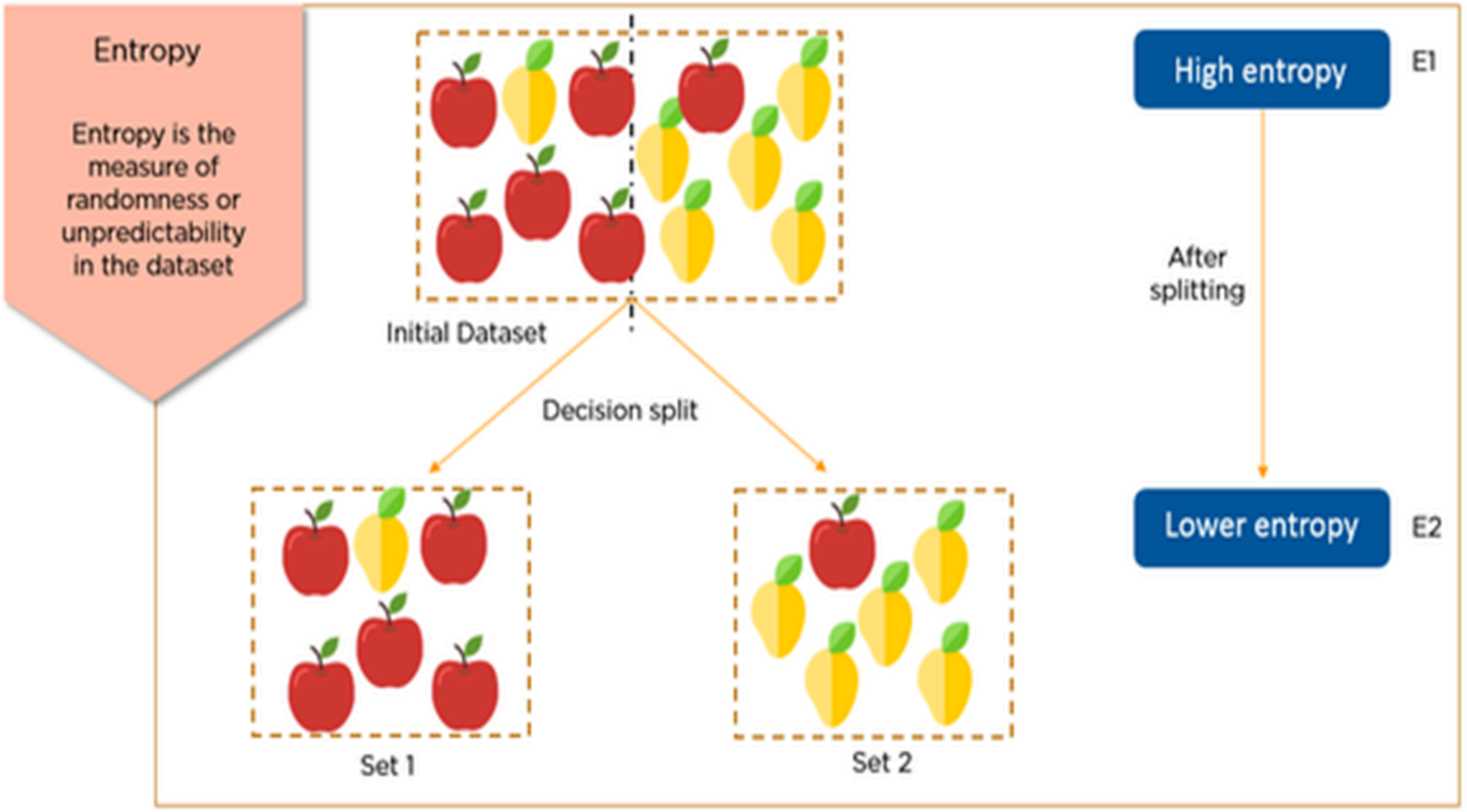
Harder to predict
(Higher entropy)

Easier to predict
(Lower entropy)

Entropy is the measure of randomness in a dataset.

Aim of DT – split the data in a way that the entropy in the data decreases -> easier to make predictions

ENTROPY



- Initial Dataset – all mixed up
- Split – less random -> entropy decreases

INFORMATION GAIN

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

$$Gain(A) = I(p,n) - E(A)$$

$$I(p,n) = -\left(\frac{p}{p+n}\log_2\frac{p}{p+n}\right) - \left(\frac{n}{p+n}\log_2\frac{n}{p+n}\right)$$

$$E(A) = \sum_{i=1}^{v}\frac{p_i + n_i}{p+n} I(p_i, n_i)$$

EXAMPLE

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Attribute Selection by Information Gain Computation

- Class P: buys_computer = "yes"
- Class N: buys_computer = "no"
- $I(p,n) = I(9,5) = 0.940$
- Compute the entropy for *age*:

$$I(p,n) = -\left(\frac{p}{p+n}\log_2\frac{p}{p+n}\right) - \left(\frac{n}{p+n}\log_2\frac{n}{p+n}\right)$$

age	p_i	n_i	$I(p_i, n_i)$
<=30	2	3	0.971
30…40	4	0	0
>40	3	2	0.971

$$E(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p+n} I(p_i, n_i)$$

$$E(age) = \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0) + \frac{5}{14} I(3,2) = 0.69$$

$$Gain(A) = I(p,n) - E(A)$$

$$Gain(age) = I(p,n) - E(age)$$
$$= 0.940 - 0.69 = 0.25$$

$$Gain(income) = 0.029$$
$$Gain(student) = 0.151$$
$$Gain(credit\_rating) = 0.048$$

EXAMPLE

```
pip install scikit-learn
from sklearn.tree import DecisionTreeClassifier
...
...
dtree = DecisionTreeClassifier()
dtree.fit(X_train, y_train)
predictions = dtree.predict(X_test)
```

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Attribute Selection by Information Gain Computation

- Class P: buys_computer = “yes”
- Class N: buys_computer = “no”
- $I(p,n) = I(9,5) = 0.940$
- Compute the entropy for *age*:

$$I(p,n) = -\left(\frac{p}{p+n}\log_2\frac{p}{p+n}\right) - \left(\frac{n}{p+n}\log_2\frac{n}{p+n}\right)$$

age	p_i	n_i	$I(p_i, n_i)$
<=30	2	3	0.971
30…40	4	0	0
>40	3	2	0.971

$$E(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p+n} I(p_i, n_i)$$

$$E(age) = \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0) + \frac{5}{14} I(3,2) = 0.69$$

$$Gain(A) = I(p,n) - E(A)$$

$$Gain(age) = I(p,n) - E(age) = 0.940 - 0.69 = 0.25$$

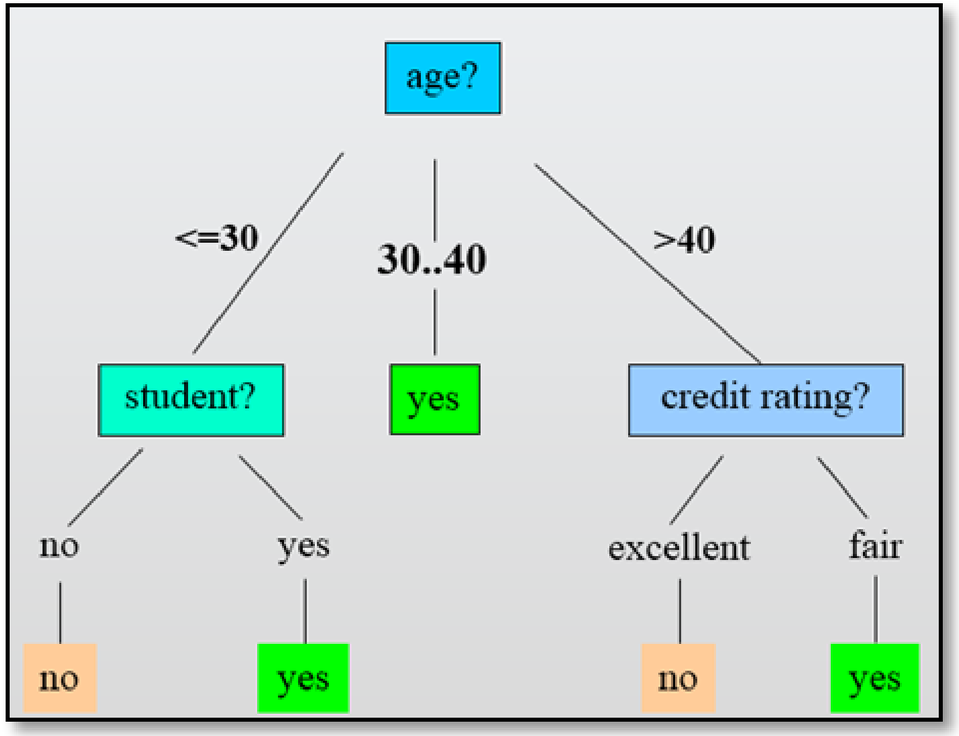
Better Information gain then the others, so it is selected for next branch

$$Gain(income) = 0.029$$

$$Gain(student) = 0.151$$

$$Gain(credit\_rating) = 0.048$$

EXTRACTING CLASSIFICATION RULES FROM TREES



- **Represent the knowledge in the form of IF-THEN rules**
- One rule is created for each path from the root to a leaf
- Each attribute-value pair along a path forms a conjunction
- The **leaf node holds the class prediction**
- Rules are easier for humans to understand

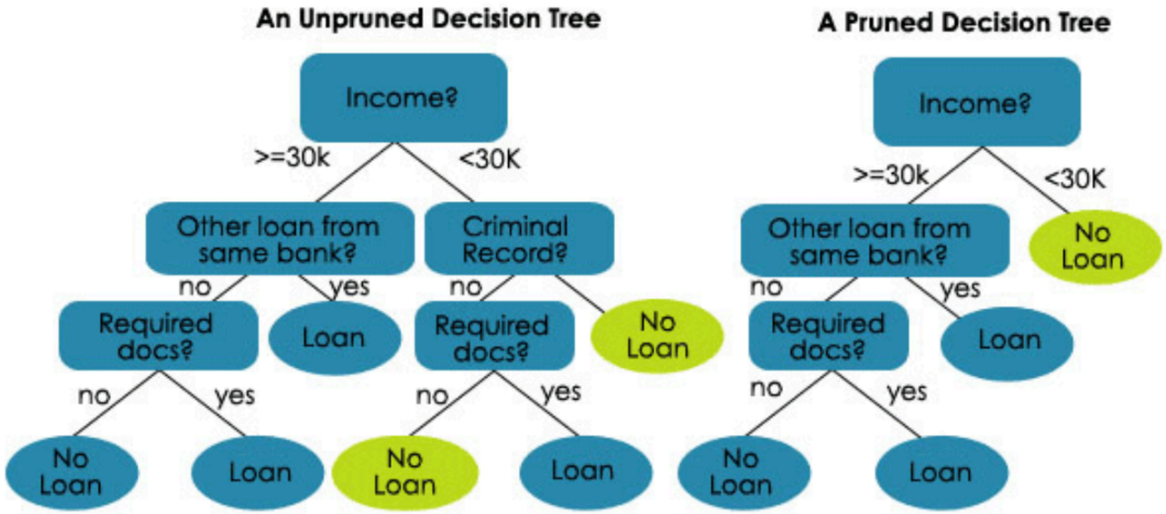
Example

IF *age* = "<=30" AND *student* = "*no*" THEN *buys_computer* = "*no*"
IF *age* = "<=30" AND *student* = "*yes*" THEN *buys_computer* = "*yes*"
IF *age* = "31…40" THEN *buys_computer* = "*yes*"
IF *age* = ">40" AND *credit_rating* = "*excellent*" THEN *buys_computer* =
IF *age* = ">40" AND *credit_rating* = "*fair*" THEN *buys_computer* = "*no*"

AVOID OVERFITTING IN CLASSIFICATION

- The generated tree may overfit the training data
  - Too many branches, some may reflect anomalies due to noise or outliers
  - Result is in poor accuracy for unseen samples

- Two approaches to avoid overfitting
  - **Prepruning:** Halt tree construction early—do not split a node if this would result in the goodness measure falling below a threshold
    - Difficult to choose an appropriate threshold
  - **Postpruning:** Remove branches from a "fully grown" tree—get a sequence of progressively pruned trees
    - Use a set of data different from the training data to decide which is the "best pruned tree"

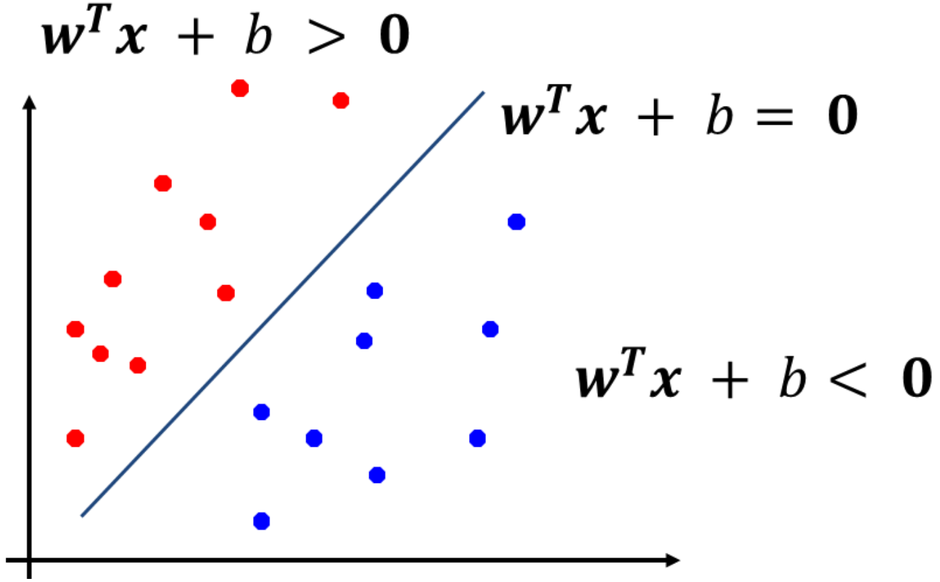


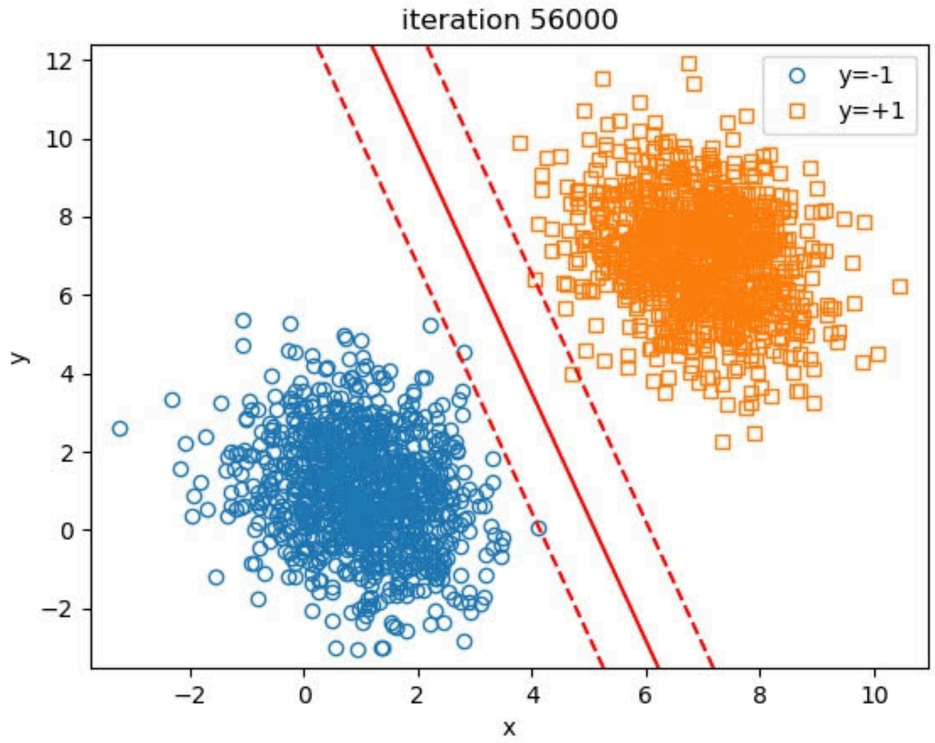
https://medium.com/nerd-for-tech/overfitting-and-pruning-in-decision-trees-improving-models-accuracy-fdbe9ecd1160

SUPPORT VECTOR MACHINE

Support Vector Machines find the “best” hyperplane that separates the two sets of points.

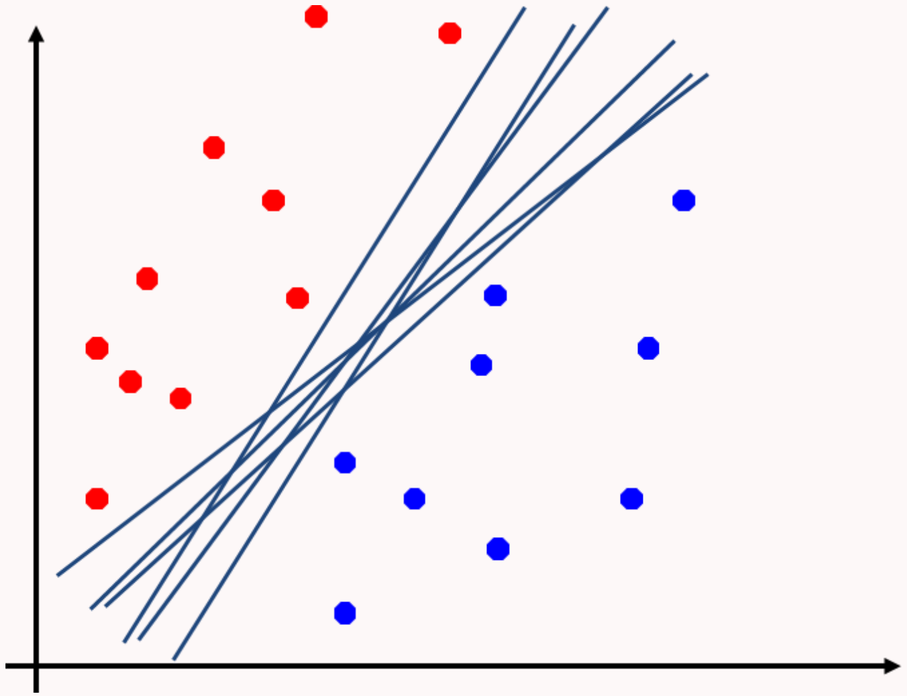
$$y(\boldsymbol{x}) = sign(\boldsymbol{w}^T\boldsymbol{x} + b)$$





SUPPORT VECTOR MACHINE

Which one is the best?

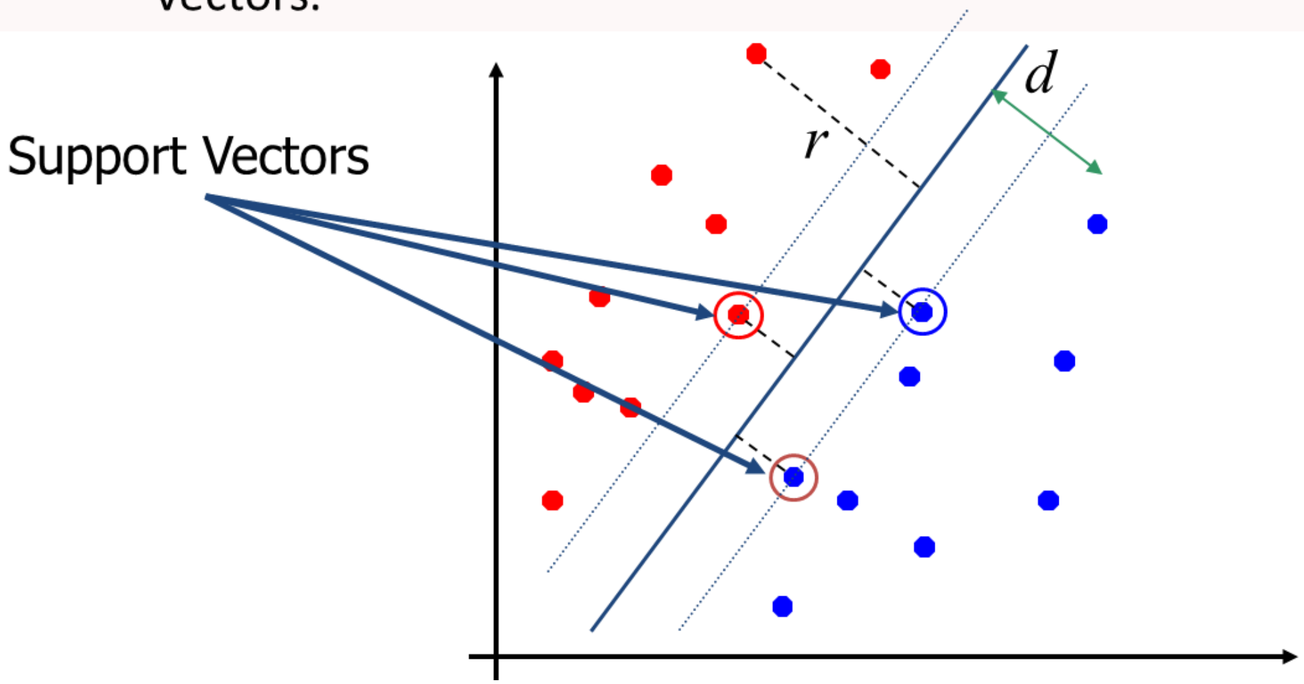


SUPPORT VECTOR MACHINE

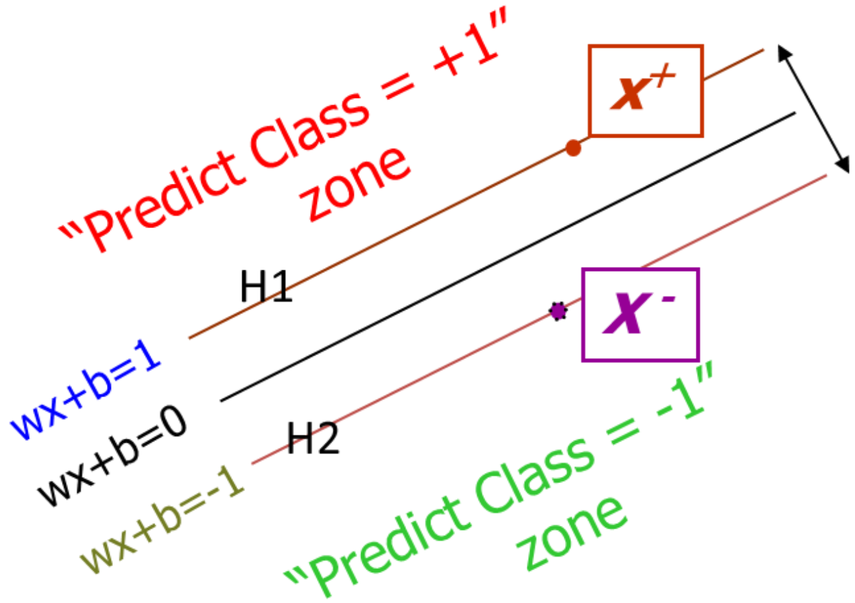
Classifier margin

- Distance from example $\mathbf{x}_i$ to the separator is
- Examples closest to the hyperplane are ***support vectors***.
- ***Margin*** ρ of the separator is the distance between support vectors.

$$r = \frac{\mathbf{w}^T \mathbf{x}_i + b}{\|\mathbf{w}\|}$$



SUPPORT VECTOR MACHINE



What we know:

- $\boldsymbol{w^T \cdot x^+ + b \geq +1}$
- $\boldsymbol{w^T \cdot x^- + b \leq -1}$
- $\boldsymbol{w^T \cdot (x^+ - x^-) = 2}$

$\boldsymbol{\rho}$=Margin Width

$$\rho = \frac{(x^+ - x^-) \cdot w^T}{||w||} = \frac{2}{||w||}$$

In order to maximize the margin, we need to minimize ||w||. With the condition that there are no datapoints between H1 and H2:

w•x_i+b ≥ +1 when y_i =+1
w•x_i+b ≤ -1 when y_i =-1

Can be combined into

$$\boldsymbol{y_i(w \cdot x_i + b) \geq 1}$$

SUPPORT VECTOR MACHINE

- Maximising the distance is the same as minimising $\frac{1}{2}\boldsymbol{w}\cdot\boldsymbol{w}$
- Subject to $\boldsymbol{y_i(w\cdot x_i+b)\geq 1}$
- If we introduce Lagrange multipliers the problem becomes

$$\frac{1}{2}\boldsymbol{w}\cdot\boldsymbol{w}-\sum_{1}^{N}\boldsymbol{\alpha_i(y_i(w\cdot x_i+b)-1)}$$

SUPPORT VECTOR MACHINE

- Minimise wrt w and b
  Maximise wrt α_i
  Some math gymnastics gives

$$\sum_{1}^{N} \boldsymbol{\alpha_i y_i x_i} = \boldsymbol{w}$$

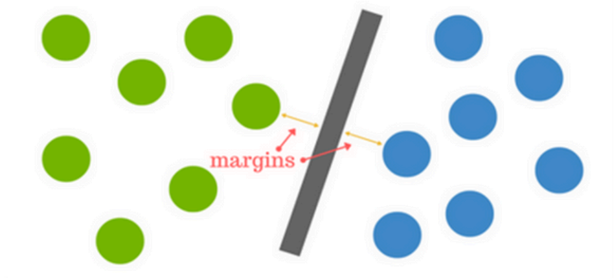
$$\sum_{1}^{N} \boldsymbol{\alpha_i y_i} = \boldsymbol{0}$$

- The hyperplane is determined by very few data points
  i.e. Most of the α_i are zero
- To classify a new data point:
  - Where the α_i are non-zero
  - Only have to calculate the support vectors

$$y(\boldsymbol{x}) = sign(\boldsymbol{w^T x} + b)$$

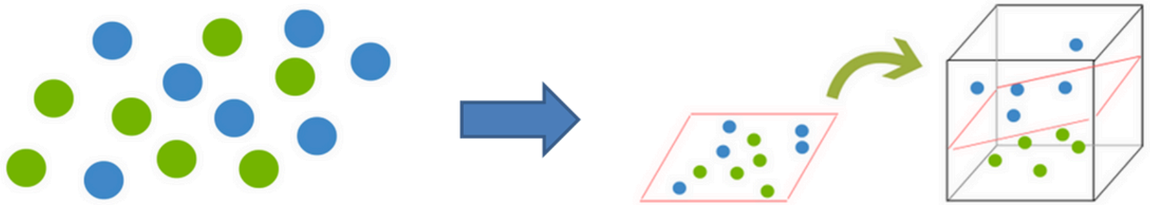
$$y(\boldsymbol{x}) = sign(\sum_{1}^{N} (\boldsymbol{\alpha_i y_i x . x_i} + b))$$

SUPPORT VECTOR MACHINE



Goal

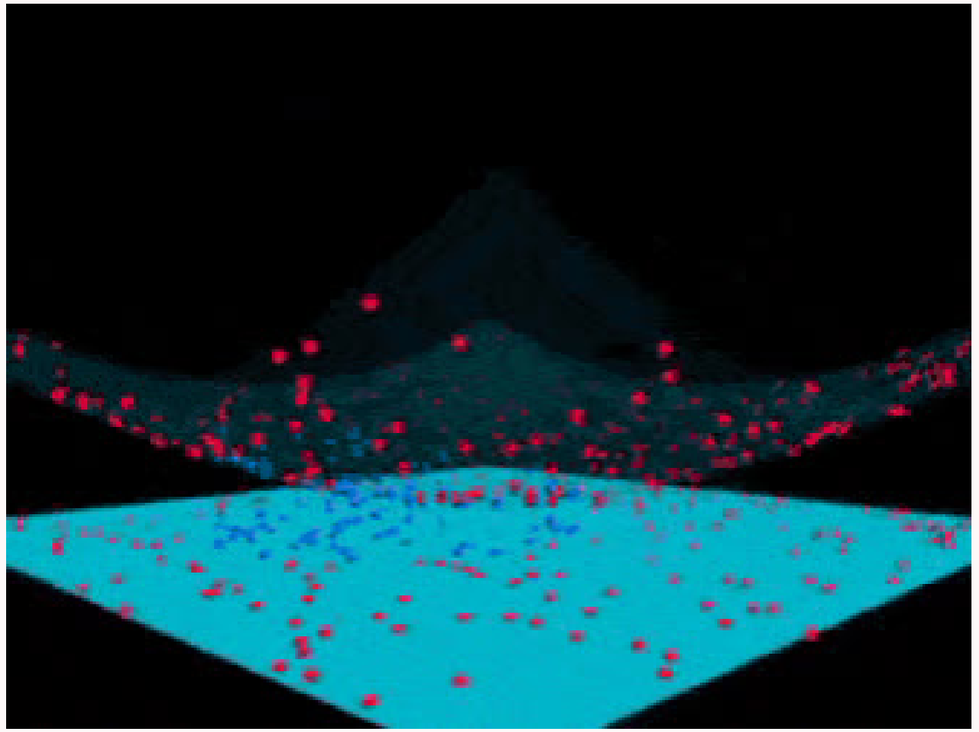
but what happens when there is no clear hyperplane?



move away from a 2d view of the data to a 3d view.

```
pip install scikit-learn
from sklearn.svm import SVC
...
...
svm_classifier = SVC(kernel='linear')
svm_classifier.fit(X_train, y_train)
predictions = svm_classifier.predict(X_test)

w = clf.coef_[0]
b = clf.intercept_[0]
```



NAÏVE BAYESIAN CLASSIFICATION

- **Probabilistic learning**: Calculate explicit probabilities for hypothesis, among the most practical approaches to certain types of learning problems.
- **Incremental**: Each training example can incrementally increase/decrease the probability that a hypothesis is correct. Prior knowledge can be combined with observed data.
- **Probabilistic prediction**: Predict multiple hypotheses, weighted by their probabilities.
- **Standard**: Even when Bayesian methods are computationally intractable, they can provide a standard of optimal decision making against which other methods can be measured.

NAÏVE BAYESIAN CLASSIFICATION

- Given training data $\mathcal{D}$, *posteriori probability of a hypothesis* h, $P(h|\mathcal{D})$ follows the Bayes theorem

$$P(h|\mathcal{D}) = \frac{P(\mathcal{D}|h)P(h)}{P(\mathcal{D})}$$

- MAP (maximum posteriori) hypothesis

$$h_{MAP} \equiv \arg\max_{h \in H} P(h|\mathcal{D}) = \arg\max_{h \in H} P(\mathcal{D}|h)P(h)$$

- **Practical difficulty: require initial knowledge of many probabilities, significant computational cost**

NAÏVE BAYESIAN CLASSIFICATION

- The classification problem may be formalized using a-posteriori probabilities:

$P(C|X)$ = prob. that the sample tuple $X = < A_1, \dots, A_k >$ is of class C.

- E.g. $P(class = N \mid outlook = sunny, windy = true, \dots)$

- **Idea: assign to sample X the class label C such that $P(C|X)$ is maximal**

NAÏVE BAYESIAN CLASSIFICATION

Bayes theorem:

$$P(C|X) = P(X|C) \cdot P(C) \,/\, P(X)$$

- $P(X)$ is constant for all classes
- $P(C)$ = relative frequency of class C samples
- C such that $P(C|X)$ is maximum =

  C such that $P(X|C) \cdot P(C)$ is maximum
- Naïve assumption: attribute independence $P(A_1, \ldots, A_k|C) = P(A_1|C) \cdot \cdots \cdot P(A_k|C)$

PLAY-TENNIS EXAMPLE: ESTIMATING P(XI|C)

Example of training data: Play tennis? (P-positive or N-negative)

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

An unseen sample X = <rain, hot, high, false>. Predict $P(C|X)$

PLAY-TENNIS EXAMPLE: ESTIMATING P(XI|C)

Bayes theorem:

$$P(C|X) = P(X|C) \cdot P(C) / P(X)$$

- $P(X)$ is constant for all classes
- $P(C)$ = relative frequency of class C samples
- C such that $P(C|X)$ is maximum =
  C such that $P(X|C) \cdot P(C)$ is maximum
- Naïve assumption: attribute independence $P(A_1, \dots, A_k|C) = P(A_1|C) \cdot \dots \cdot P(A_k|C)$

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

P(p) = 9/14
P(n) = 5/14

outlook	
P(sunny|p) = 2/9	**P(sunny|n) = 3/5**
P(overcast|p) = 4/9	**P(overcast|n) = 0**
P(rain|p) = 3/9	**P(rain|n) = 2/5**
temperature	
P(hot|p) = 2/9	**P(hot|n) = 2/5**
P(mild|p) = 4/9	**P(mild|n) = 2/5**
P(cool|p) = 3/9	**P(cool|n) = 1/5**
humidity	
P(high|p) = 3/9	**P(high|n) = 4/5**
P(normal|p) = 6/9	**P(normal|n) = 2/5**
windy	
P(true|p) = 3/9	**P(true|n) = 3/5**
P(false|p) = 6/9	**P(false|n) = 2/5**

An unseen sample ***X*** = **<rain, hot, high, false>.** Predict $P(C|X)$

PLAY-TENNIS EXAMPLE: ESTIMATING P(XI|C)

```
pip install scikit-learn
from sklearn.naive_bayes import MultinomialNB
...
...
model = MultinomialNB()
model.fit(X_train, y_train)
predictions = model.predict(X_test)
```

An unseen sample X = <rain, hot, high, false>. Predict $P(C|X)$

$$P(X|p) \cdot P(p) =$$
$$P(rain|p) \cdot P(hot|p) \cdot P(high|p) \cdot P(false|p) \cdot P(p)$$
$$= 3/9 \cdot 2/9 \cdot 3/9 \cdot 6/9 \cdot 9/14 = 0.010582$$

$$P(X|n) \cdot P(n)$$
$$= P(rain|n) \cdot P(hot|n) \cdot P(high|n) \cdot P(false|n) \cdot P(n)$$
$$= 2/5 \cdot 2/5 \cdot 4/5 \cdot 2/5 \cdot 5/14 = \mathbf{0.018286}$$

Conclusion: Sample X is classified in class n (don't play)

$* P(h|\mathcal{D}) = \frac{P(\mathcal{D}|h)P(h)}{P(\mathcal{D})}$ or $P(C|X) = \frac{P(X|C)P(C)}{P(X)}$

$P(\mathcal{D})$ is a constant = 0.029

P(rain) x P(hot) X P(high) x P(false) = 5/14*4/14*7/14*8/14
= 0.357*0.2357*0.5*0.5714
= 0.029

P(p) = 9/14
P(n) = 5/14

outlook	
P(sunny|p) = 2/9	P(sunny|n) = 3/5
P(overcast|p) = 4/9	P(overcast|n) = 0
P(rain|p) = 3/9	P(rain|n) = 2/5
temperature	
P(hot|p) = 2/9	P(hot|n) = 2/5
P(mild|p) = 4/9	P(mild|n) = 2/5
P(cool|p) = 3/9	P(cool|n) = 1/5
humidity	
P(high|p) = 3/9	P(high|n) = 4/5
P(normal|p) = 6/9	P(normal|n) = 2/5
windy	
P(true|p) = 3/9	P(true|n) = 3/5
P(false|p) = 6/9	P(false|n) = 2/5

ASSESSING CLASSIFIER & REGRESSION

Classification

CONFUSION MATRIX (CM)	CROSS-VALIDATION TECHNIQUES
RECEIVER OPERATING CHARACTERISTIC CURVE (ROC) & AREA UNDER THE ROC CURVE (AUC)	PRECISION-RECALL CURVE (PR CURVE)

Regression

MEAN SQUARE ERROR(MSE)	MEAN ABSOLUTE ERROR (MAE)
R-SQUARED (R2)	MEDIAN ABSOLUTE ERROR (MDAE)

CONFUSION MATRIX

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known.

n=165	**Predicted: NO**	**Predicted: YES**
Actual: NO	50	10
Actual: YES	5	100

```
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming you have the true labels and predicted labels as follows:
# True labels
y_true = ['NO']*55 + ['YES']*105
# Predicted labels
y_pred = ['NO']*50 + ['YES']*10 + ['NO']*5 + ['YES']*95

# Generate the confusion matrix
cm = confusion_matrix(y_true, y_pred, labels=['NO', 'YES'])

# Plot the confusion matrix using Seaborn's heatmap
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['NO', 'YES'], yticklabels=
['NO', 'YES'])
plt.ylabel('Actual')
plt.xlabel('Predicted')
plt.title('Confusion Matrix')
plt.show()
```

CONFUSION MATRIX

Assume we would like to predict “Yes”:

n=165	**Predicted: NO**	**Predicted: YES**
Actual: NO	True Negative 50	False Positive 10
Actual: YES	False Negative 5	True Positive 100

ACCURACY, PRECISION, AND RECALL

n=165	Predicted: NO	Predicted: YES
Actual: NO	True Negative 50	False Positive 10
Actual: YES	False Negative 5	True Positive 100

Accuracy: Overall, how often is the classifier correct?

- (TP+TN)/total = (100+50)/165 = 0.91

Precision: When it predicts yes, how often is it correct?

- TP/predicted yes = 100/110 = 0.91

Recall: When it's actually yes, how often does it predict yes?

- TP/actual yes = 100/105 = 0.95

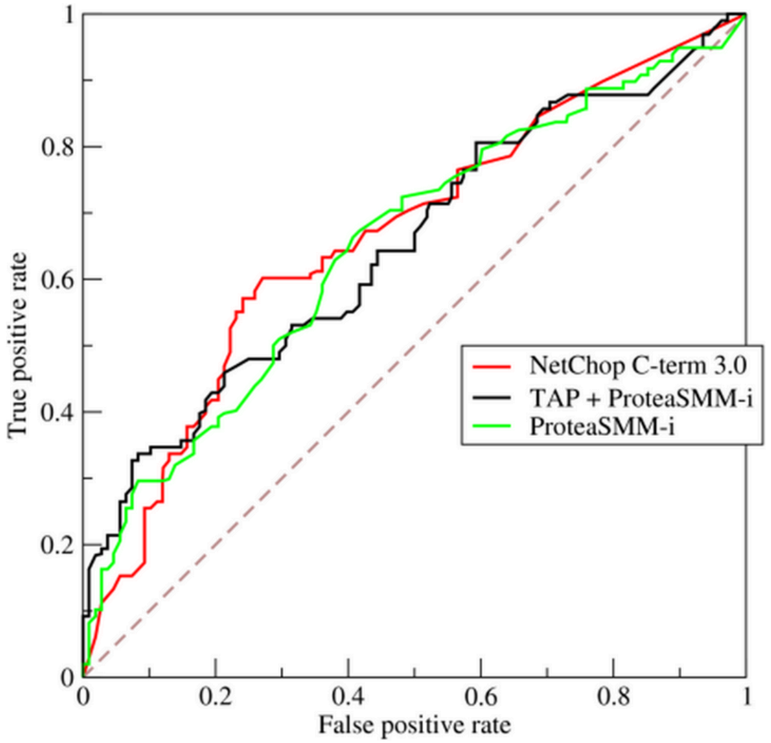
TRY THIS

- Suppose a computer program for recognizing dogs in scenes from a video identifies 7 dogs in a scene containing 9 dogs and 9 cats.
- If 4 of the identifications are correct, but 3 are actually cats, the program's precision is 4/7 while its recall is 4/9.

		Predicted	
		Dog	**cat**
Actual	**dog**	4	5
	cat	3	6

ROC CURVES AND AREA UNDER THE CURVE

- ROC is the most commonly used way to visualize the performance of a binary classifier.
- Area Under the Curve (AUC) is (arguably) the best way to summarize its performance in a single number.

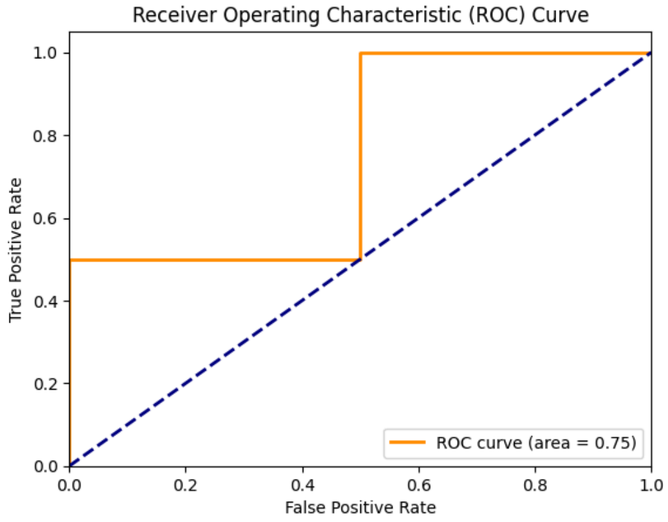



```
import numpy as np
from sklearn.metrics import roc_curve, roc_auc_score
import matplotlib.pyplot as plt

# True binary labels (0 or 1)
y_true = np.array([0, 0, 1, 1])

# Predicted probabilities of the positive class
y_scores = np.array([0.1, 0.4, 0.35, 0.8])

fpr, tpr, thresholds = roc_curve(y_true, y_scores)
auc = roc_auc_score(y_true, y_scores)
print(f'AUC: {auc:.2f}')
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
```



CROSS VALIDATION

- Labelled data sets are difficult to get
- Leave one out cross validation
  - Leave one example out and test the classification error on that one
  - Iterate through the data set
  - Compute the average classification error
- K-fold cross validation
  - Split the data set in to K sub-sets, leave one out
  - 10 fold cross validation common

```
from sklearn.model_selection import KFold
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression  # or any other model
import numpy as np
# X = features
# y = labels
kf = KFold(n_splits=5, random_state=None, shuffle=False)
model = LogisticRegression()
accuracies = []

for train_index, test_index in kf.split(X):
        X_train, X_test = X[train_index], X[test_index]
        y_train, y_test = y[train_index], y[test_index]
        model.fit(X_train, y_train)
        predictions = model.predict(X_test)
        accuracy = accuracy_score(y_test, predictions)
        accuracies.append(accuracy)
average_accuracy = np.mean(accuracies)
print(f"Average Accuracy: {average_accuracy}")
```

THE END

NEXT LECTURE

Machine Learning (Unsupervised)