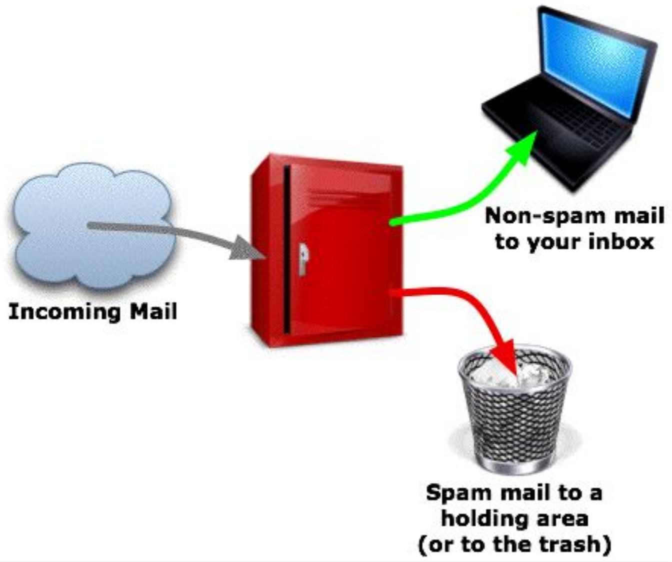
Supervised Learning

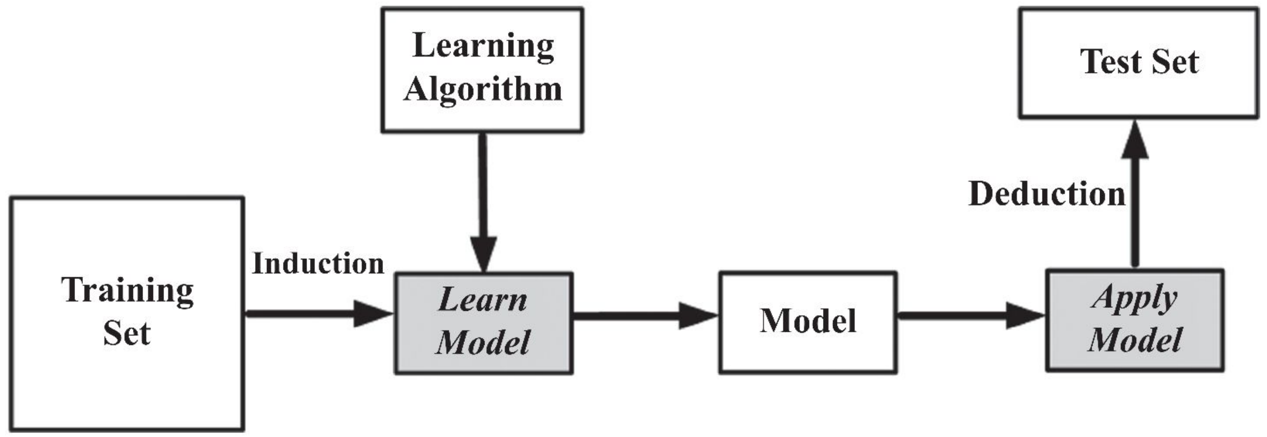
Classification Example

Learning patterns from labeled data and classify new data with labels (categories), e.g.:

- Classify a Twitter user a “bot” or “human”.
- Classify a Facebook post as “spam”.
- Decide if we should play golf.
- Classify an e-mail as "legitimate" or "spam"

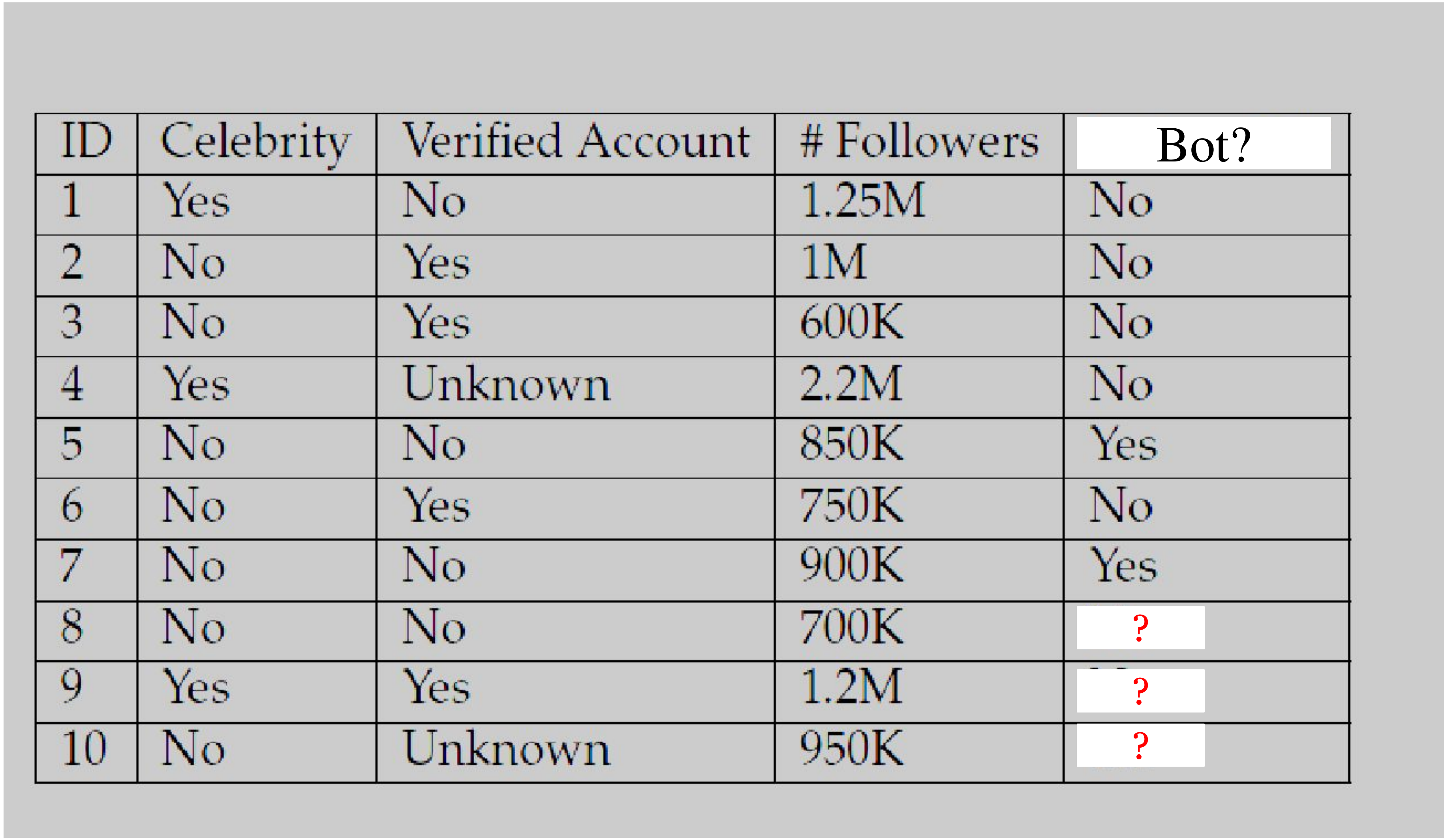

Incoming Mail
Non-spam mail to your inbox
Spam mail to a holding area (or to the trash)

Supervised Learning: The Process



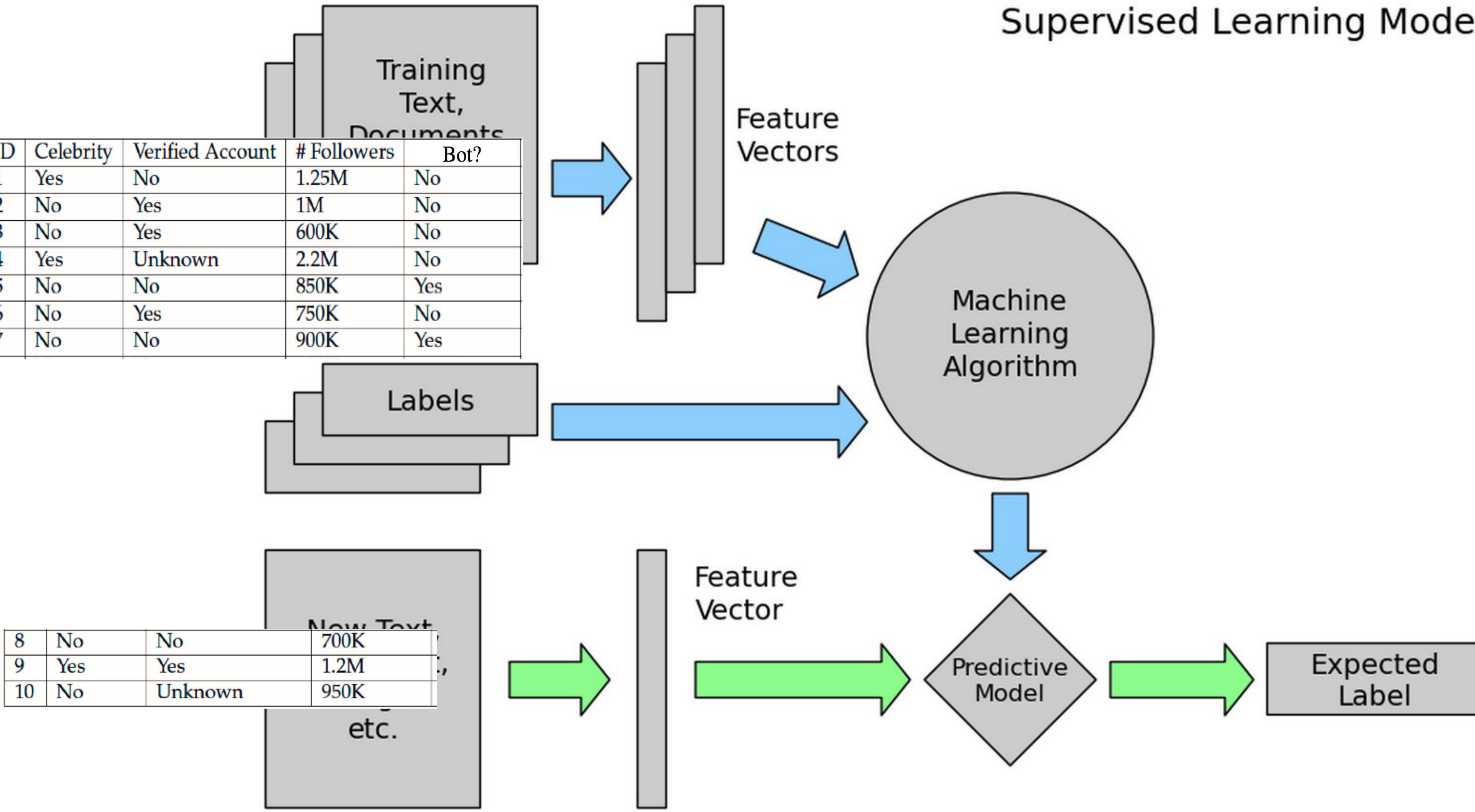
- We are given a set of labeled examples
- These examples are records/instances in the format (**x,** y) where **x** is a vector and y is the class attribute, commonly a scalar
- The supervised learning task is to build model that maps **x** to y (find a mapping *m* such that m(**x**) = y)
- Given an unlabeled instances (x',?), we compute m(x')
  - E.g., spam/non-spam prediction

A Twitter Example



ID	Celebrity	Verified Account	# Followers	Bot?
1	Yes	No	1.25M	No
2	No	Yes	1M	No
3	No	Yes	600K	No
4	Yes	Unknown	2.2M	No
5	No	No	850K	Yes
6	No	Yes	750K	No
7	No	No	900K	Yes
8	No	No	700K	?
9	Yes	Yes	1.2M	?
10	No	Unknown	950K	?

Supervised Learning: The Process



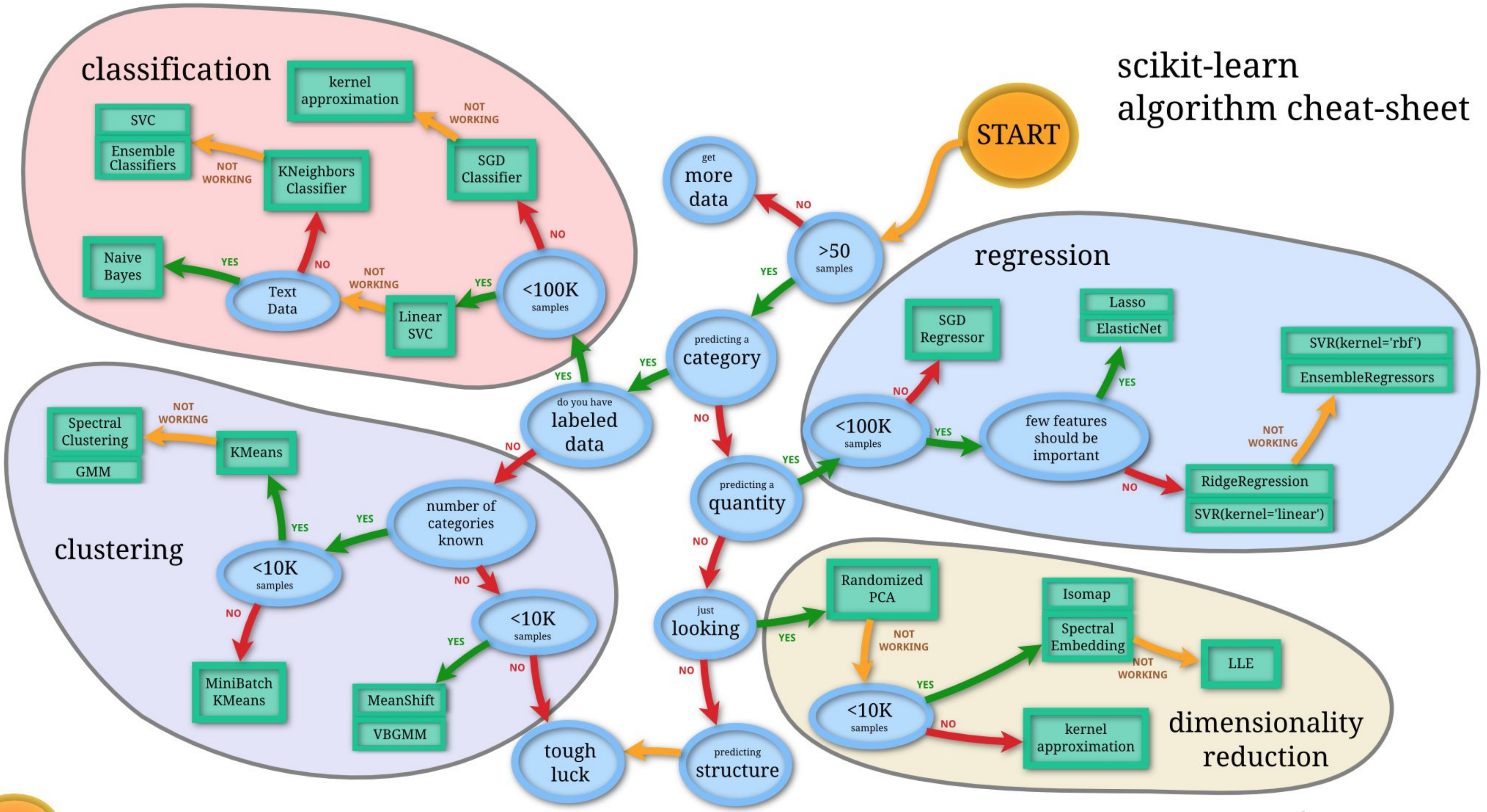
D	Celebrity	Verified Account	# Followers	Bot?
1	Yes	No	1.25M	No
2	No	Yes	1M	No
3	No	Yes	600K	No
4	Yes	Unknown	2.2M	No
5	No	No	850K	Yes
6	No	Yes	750K	No
7	No	No	900K	Yes

8	No	No	700K	
9	Yes	Yes	1.2M	
10	No	Unknown	950K	

Weka

- Tool containing many data mining algorithms.
- Allow for exploration/comparison of different data mining algorithms.

Scikit-Learn



Back

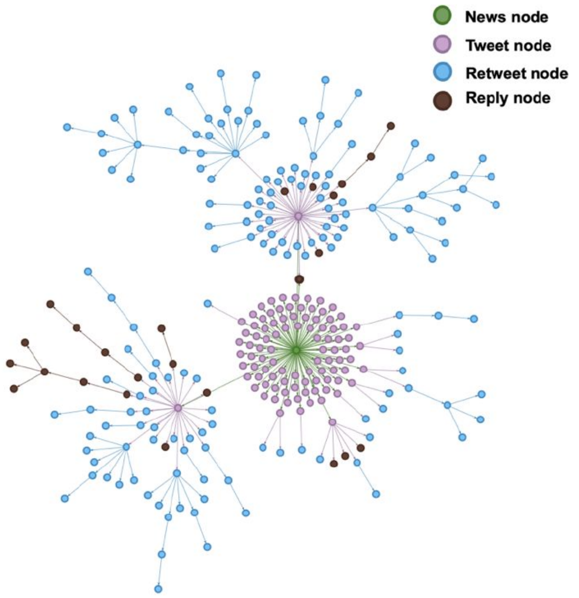
Classification: Example

- Kai Shu, Deepak Mahudeswaran, Suhang Wang, and Huan Liu. "Hierarchical Propagation Networks for Fake News Detection: Investigation and Exploitation." ICWSM 2020.

- Trying to detect fake news on Twitter.

Classification

- **Example:** Shu et. al 2020
- News spreads in propagation networks
  - Macro-level: Nodes in this network represent the **retweets** and **tweets** posting the news
  - Micro-level: Nodes in this network represent the chain of **replies**.

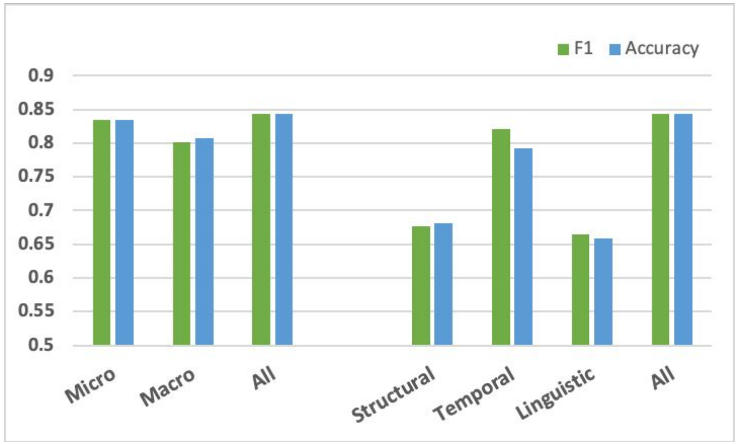


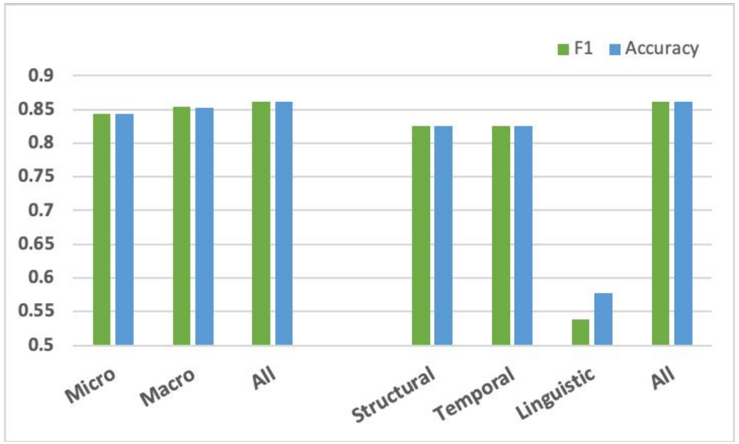
Feature extraction

- **Example:** Shu et. al 2020
- Authors hand-pick several types features
- Structural Features
  - Tree depth, Number of nodes in macro-network, Maximum Outdegree, Number of cascades, …
- Temporal Features
  - Time difference between the first tweet and the last retweets…
- Linguistic Features
  - Average sentiment, Average sentiment of first level replies, …

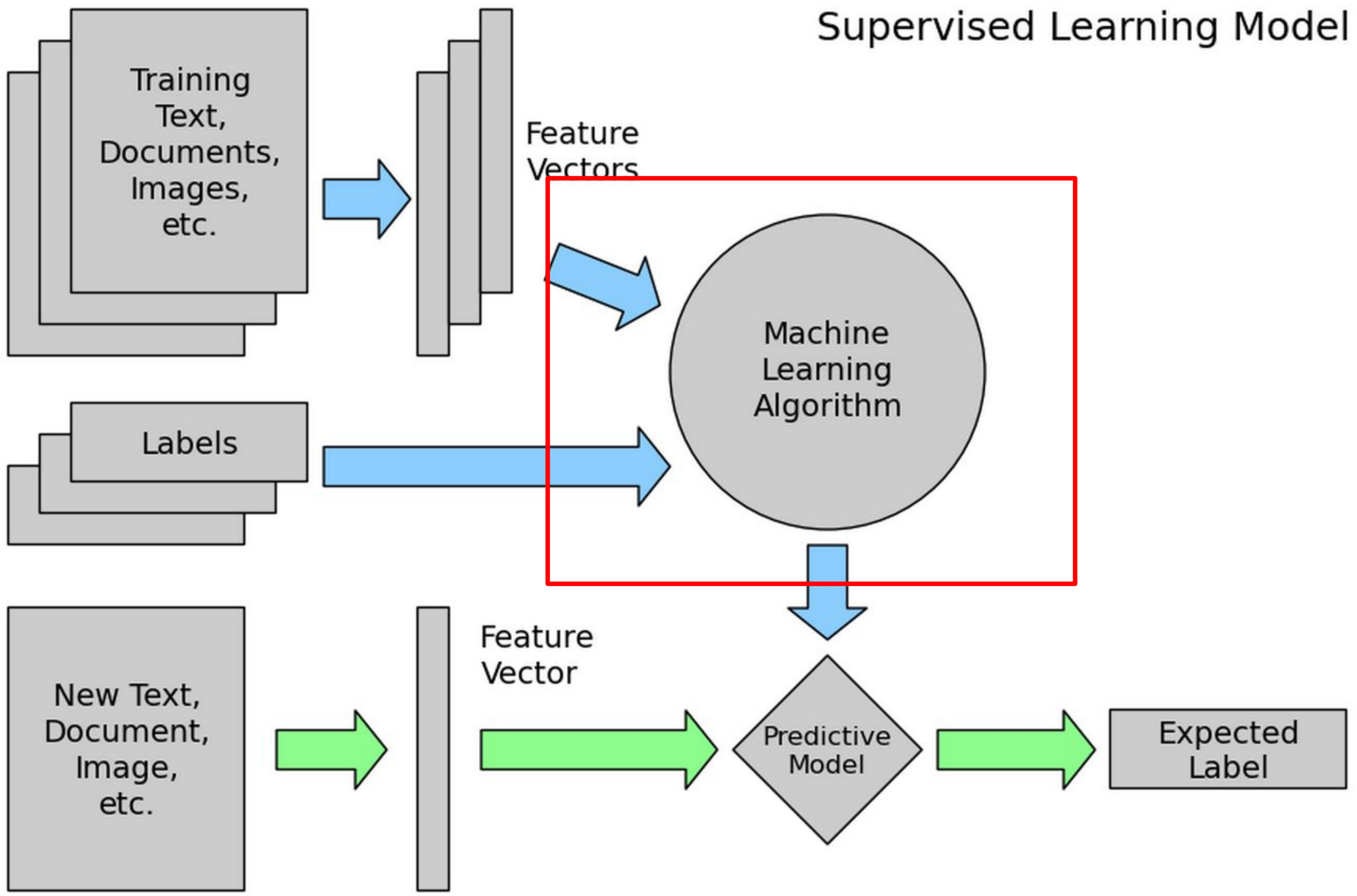
Classification Results

- Micro and Macro level network features performs similarly well
- Temporal features perform better than other features





Classification



Supervised Learning Model
Training Text, Documents, Images, etc.
Feature Vectors
Machine Learning Algorithm
Labels
New Text, Document, Image, etc.
Feature Vector
Predictive Model
Expected Label

Supervised Learning Algorithms

- Classification
  - *k*-Nearest Neighbor Classifier
  - Decision tree learning
  - Naive Bayes Classifier
  - Classification with Network information
- Regression
  - Linear Regression
  - Logistic Regression

k-Nearest Neighbor Classifier

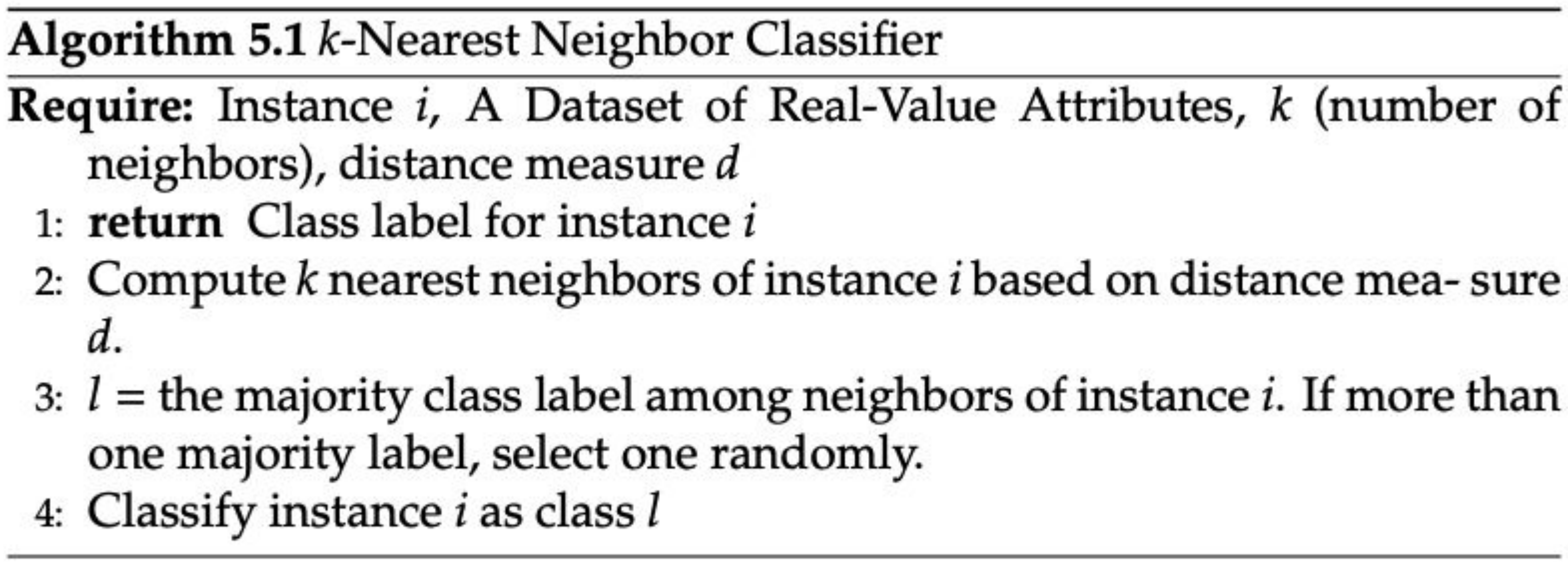
- *k*-Nearest Neighbors or kNN, as the name suggests, utilizes the neighbors of an instance to perform classification.

- To determine the neighbors of an instance, we need to measure its distance to all other instances based on some distance metric. Commonly, Euclidean distance is employed

- The instance being classified is assigned the label (class attribute value) that the majority of its *k* neighbors are assigned

- When $k = 1$, the closest neighbor's label is used as the predicted label for the instance being classified

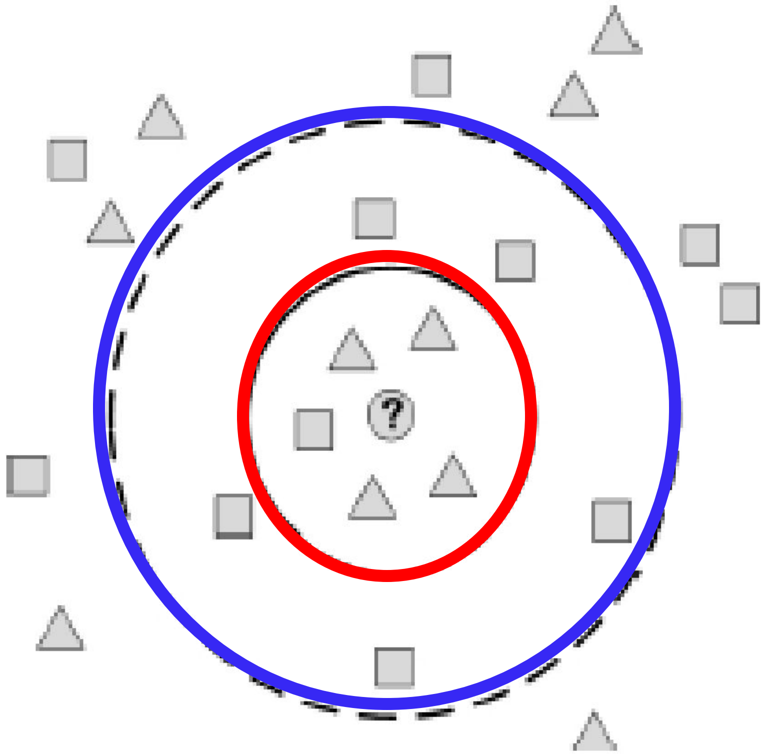
K-NN: Algorithm



Algorithm 5.1 *k*-Nearest Neighbor Classifier

```
Require: Instance $i$, A Dataset of Real-Value Attributes, $k$ (number of
    neighbors), distance measure $d$
1: return Class label for instance $i$
2: Compute $k$ nearest neighbors of instance $i$ based on distance mea- sure
   $d$.
3: $l$ = the majority class label among neighbors of instance $i$. If more than
   one majority label, select one randomly.
4: Classify instance $i$ as class $l$
```

k-NN example



- When k=5, the predicted label is: triangle
- When k=9, the predicted label is: square

Supervised Learning Algorithms

- Classification
  - *k*-Nearest Neighbor Classifier
  - Decision tree learning
  - Naive Bayes Classifier
  - Classification with Network information
- Regression
  - Linear Regression
  - Logistic Regression

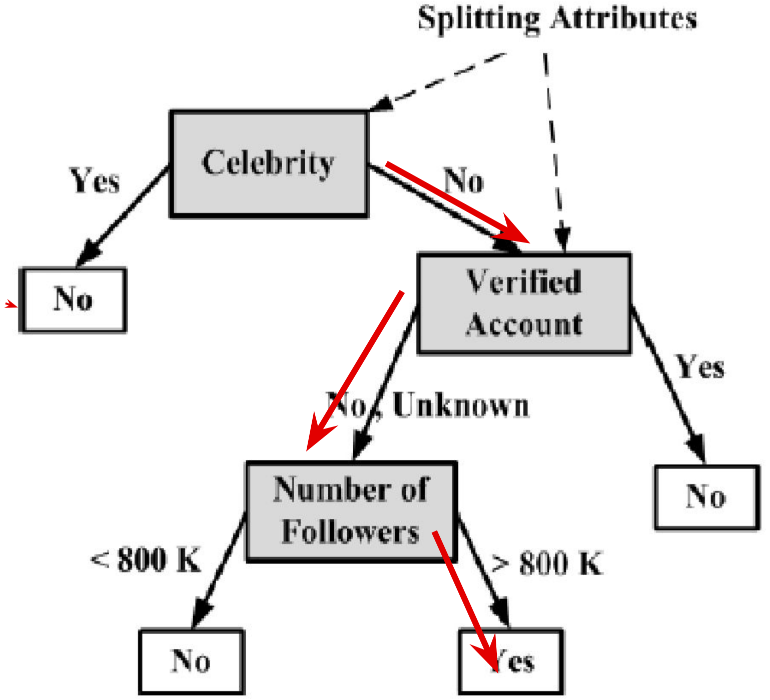
Decision Tree

- A decision tree is learned from the dataset (training data with known classes) and later applied to predict the class attribute value of new data (test data with unknown classes) where only the feature values are known

Decision Tree: Example

ID	Celebrity	Verified Account	# Followers	Bot?
1	Yes	No	1.25M	No
2	No	Yes	1M	No
3	No	Yes	600K	No
4	Yes	Unknown	2.2M	No
5	No	No	850K	Yes
6	No	Yes	750K	No
7	No	No	900K	Yes
8	No	No	700K	No
9	Yes	Yes	1.2M	No
10	No	Unknown	950K	Yes

Class
Labels



Decision Tree Construction

- Decision trees are constructed recursively from training data using a **top-down greedy approach** in which features are sequentially selected.

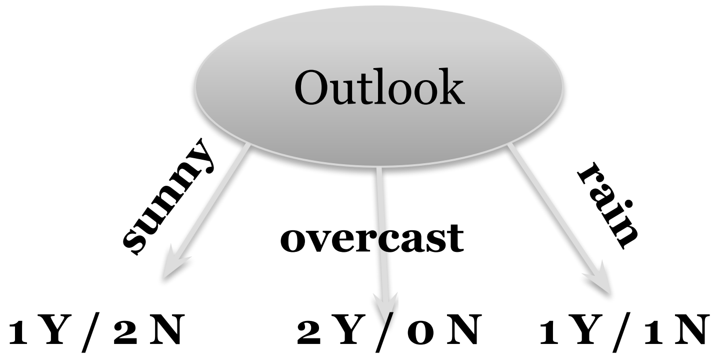
- After selecting a feature for each node, based on its values, different branches are created.

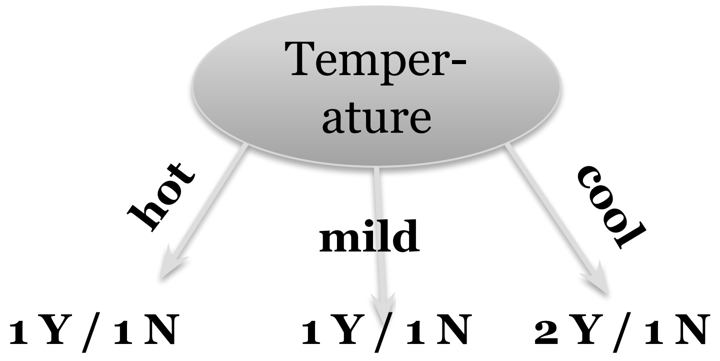
- The training set is then partitioned into subsets based on the feature values, each of which fall under the respective feature value branch; the process is continued for these subsets and other nodes.

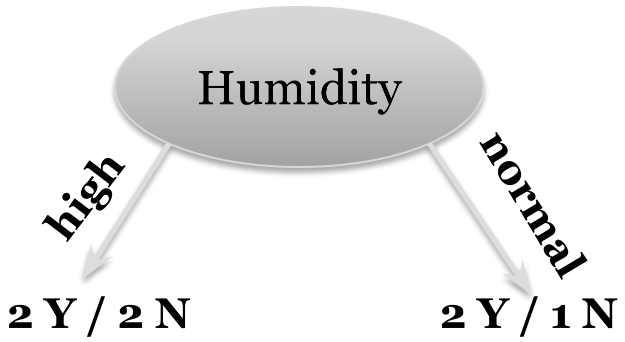
- When selecting features, we ***prefer features*** *that partition the set of instances into* ***subsets*** *that are more* ***pure***. A pure subset has instances that all have the same class attribute value.

Decision Tree Construction

No.	Outlook (O)	Temperature (T)	Humidity (H)	Play Golf (PG)
1	sunny	hot	high	N
2	sunny	mild	high	N
3	overcast	hot	high	Y
4	rain	mild	high	Y
5	sunny	cool	normal	Y
6	rain	cool	normal	N
7	overcast	cool	normal	Y







Decision Tree Construction

- To measure purity we can use entropy. Over a subset of training instances, T, with a binary class attribute (values in {+,-}), the entropy of T is defined as:

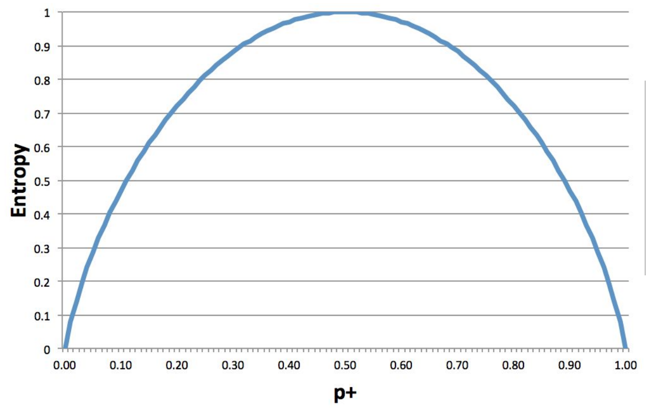
$$entropy(T) = -p_+ \log p_+ - p_- \log p_-,$$

  - p_+ is the proportion of positive examples in D
  - p_- is the proportion of negative examples in D
- We can use this to measure the quality of the leaves along a split.

Entropy is a measure of disorder

Entropy Example

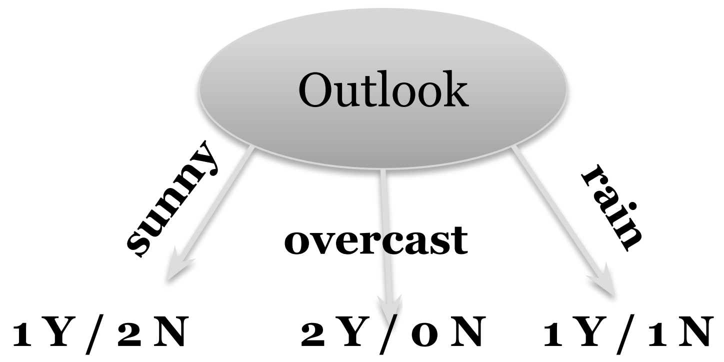
Assume there is a subset T, containing 10 instances. Seven instances have a positive class attribute value and three have a negative class attribute value [7+, 3-]. The entropy measure for subset T is:



$$entropy(T) = -\frac{7}{10}\log\frac{7}{10} - \frac{3}{10}\log\frac{3}{10} = 0.881.$$

In a pure subset, all instances have the same class attribute value and the entropy is 0.

Entropy Example



H(sunny) = $-1/3 * \log_2(1/3) - 2/3 * \log_2(2/3) \sim= 0.92$

H(overcast) = $-2/2 * \log_2(2/2) - 0/2 * \log_2(0/2) = 0$

H(rain) = $-1/2 * \log_2(1/2) - 1/2 * \log_2(1/2) = 1$

- We have measured each *value* in the attribute.
- We need to aggregate these to measure the quality of *splitting on this attribute overall*.

Information Gain

- Entropy does not tell the whole story.
- We want many items in pure sets.
- Information Gain can tell us how much our knowledge has improved after the split, i.e. our reduction in entropy

$$IG(\underbrace{A}_{\text{Attribute}}, S) = \overbrace{H(S)}^{\text{What's the pureness if I do no split the node?}} - \sum_{v \in Values(A)} \overbrace{\frac{|S_v|}{|S|}}^{\text{What's the pureness if I split based on attribute A?}} H(S_v)$$

IG(outlook, play golf) = Entropy(play golf) – Entropy(outlook, play golf)

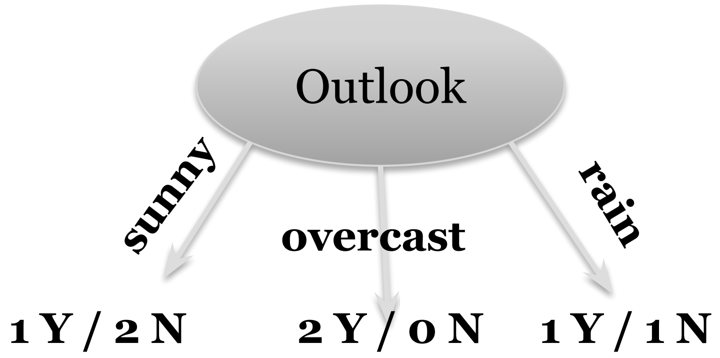
Entropy(outlook, play golf) = P(sunny)*H(outlook = sunny) + P(overcast)*H(outlook = overcast) + P(rain)*H(outlook = rain)

Information Gain

No.	Outlook (O)	Temperature (T)	Humidity (H)	Play Golf (PG)
1	sunny	hot	high	N
2	sunny	mild	high	N
3	overcast	hot	high	Y
4	rain	mild	high	Y
5	sunny	cool	normal	Y
6	rain	cool	normal	N
7	overcast	cool	normal	Y

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

$H(S) = -4/7*\log_2(4/7) - 3/7*\log_2(3/7) \sim= 0.99$

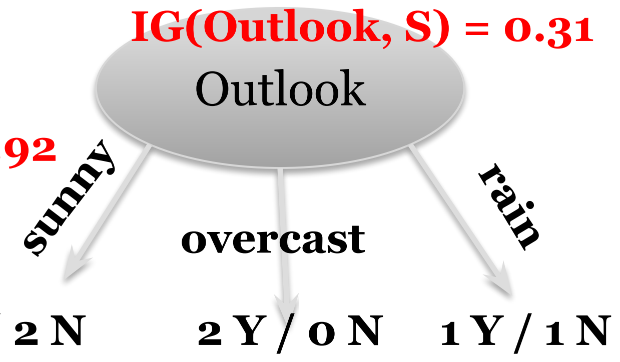


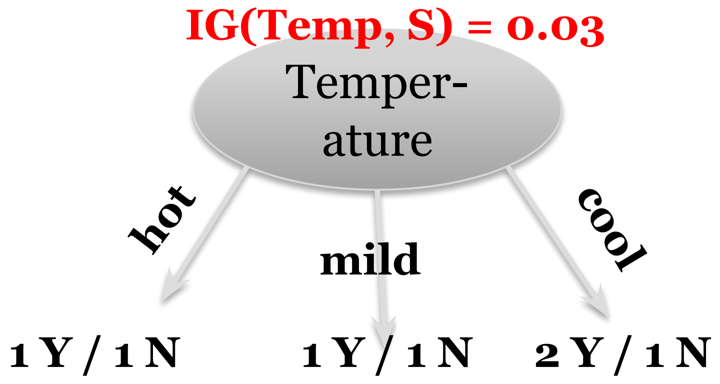
IG(A, S) = H(S) – [
3/7 * H(Outlook = Sunny) +
2/7 * H(Outlook = Overcast) +
2/7 * H(Outlook = Rain)]
= 0.99 – [3/7*.92 + 2/7*0 + 2/7*1]
= 0.31

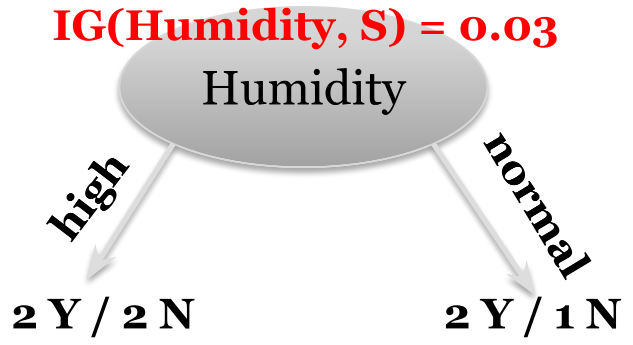
Decision Tree Construction

No.	Outlook (O)	Temperature (T)	Humidity (H)	Play Golf (PG)
1	sunny	hot	high	N
2	sunny	mild	high	N
3	overcast	hot	high	Y
4	rain	mild	high	Y
5	sunny	cool	normal	Y
6	rain	cool	normal	N
7	overcast	cool	normal	Y

H(sunny) = $-1/3 * \log_2(1/3) - 2/3 * \log_2(2/3) \approx= 0.92$



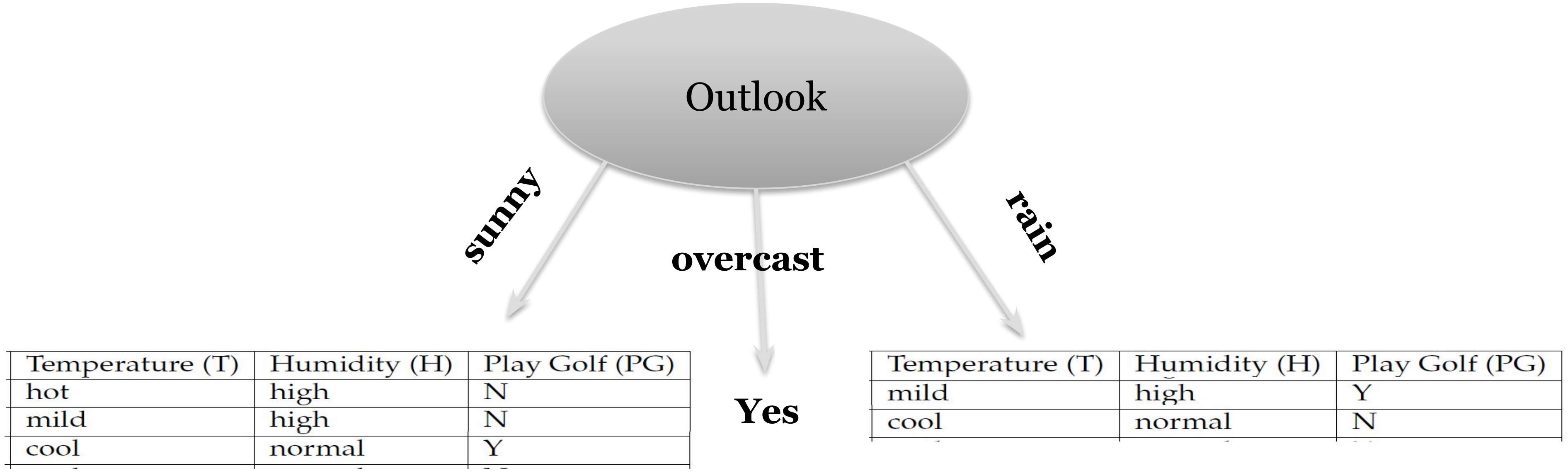




$$entropy(T) = -p_+ \log p_+ - p_- \log p_-,$$

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

Decision Tree Construction



Temperature (T)	Humidity (H)	Play Golf (PG)
hot	high	N
mild	high	N
cool	normal	Y

Temperature (T)	Humidity (H)	Play Golf (PG)
mild	high	Y
cool	normal	N

- Select the attribute with the *highest* information gain
  - Remaining subtrees are more pure.
- Repeat process recursively until all we have are leaf nodes.

Decision Tree Algorithm (ID3) [Wikipedia]

```
ID3 (Examples, Target_Attribute, Attributes)
1. Create a root node for the tree, called Root
2. If all examples are positive, Return the single-node tree Root, with label +.
3. If all examples are negative, Return the single-node tree Root, with label -.
4. If number of predicting attributes is empty, then Return the single node tree
   Root, with label = most common value of the target attribute in the
   examples.
5. Otherwise, Begin
    1. A ← The Attribute that best classifies examples.
    2. Decision Tree attribute for Root = A.
    3. For each possible value, v_i, of A,
        1. Add a new tree branch below Root, corresponding to the test A = v_i.
        2. Let Examples(v_i) be the subset of examples that have the value v_i for A
        3. If Examples(v_i) is empty. Then below this new branch add a leaf node with label = most
           common target value in the examples
        4. Else below this new branch add the subtree ID3 (Examples(v_i), Target_Attribute, Attributes –
           {A})
6. End
7. Return Root
```

Naive Bayes Classifier

- First, we should understand Bayes' Theorem.

$$P(A|B)$$

- Probability that A occurs *given that* B occurred.

$$P(A \cap B) = P(A)P(B|A)$$
$$P(A \cap B) = P(B)P(A|B)$$

$$P(B)P(A|B) = P(A)P(B|A)$$

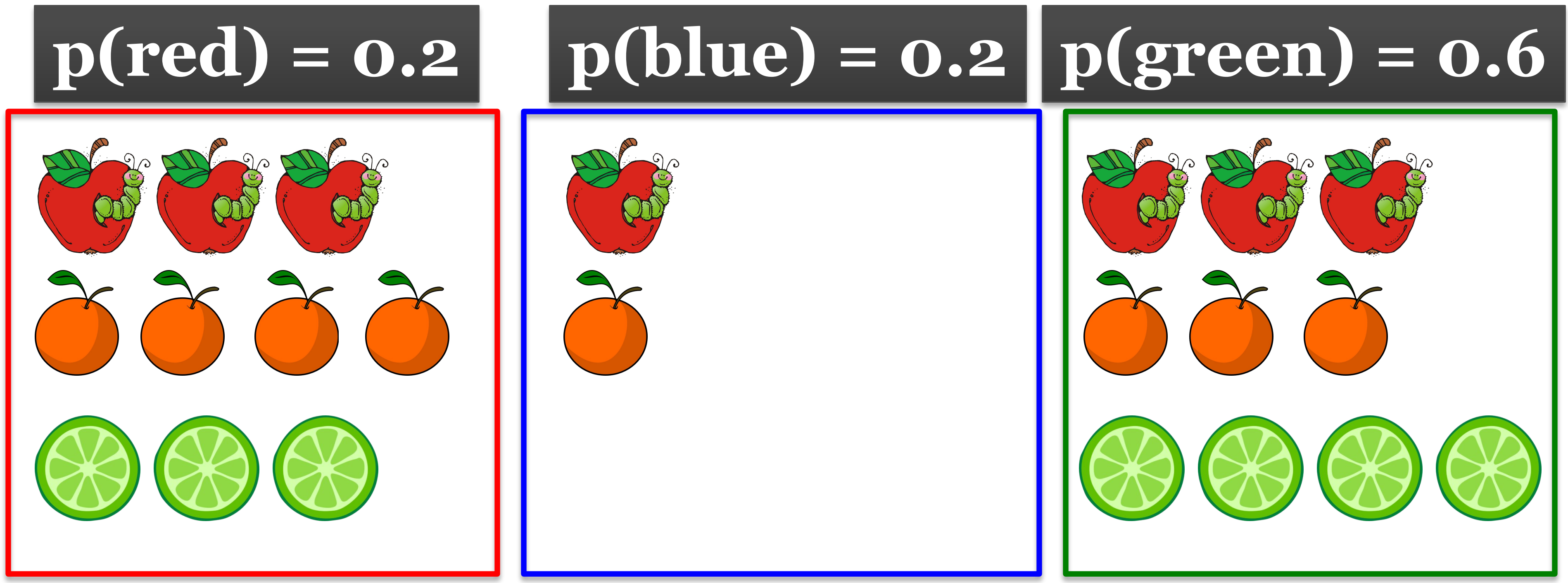
$$\boxed{P(A|B) = \frac{P(B|A)P(A)}{P(B)}}$$

Bayes Theorem Example

- Suppose you stumble across a beetle, which might be a rare subspecies of beetle, due to the pattern on its back.
  - In the rare subspecies, 98% have the pattern.
  - In the common subspecies, 5% have the pattern.
  - The rare subspecies accounts for only 0.1% of the population.
- What is the probability you have found a rare beetle?

$$P(X) = \sum_{Y} P(X|Y)P(Y)$$

Bayes Theorem Example



- What is the probability of selecting an apple?
- We have selected an *orange*. What is the probability that it came from a *green* box?

Naive Bayes Classifier

For two random variables X and Y, Bayes theorem states that,

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

class variable

the instance features

Then class attribute value for instance X

$$\arg\max_{y_i} P(y_i|X)$$

$$P(y_i|X) = \frac{P(X|y_i)P(y_i)}{P(X)}$$

We assume that features are independent given the class attribute

$$P(X|y_i) = \Pi_{j=1}^{n} P(x_j|y_i) \Longrightarrow P(y_i|X) = \frac{(\Pi_{j=1}^{n} P(x_j|y_i))P(y_i)}{P(X)}$$

NBC: An Example

No.	Outlook (O)	Temperature (T)	Humidity (H)	Play Golf (PG)
1	sunny	hot	high	N
2	sunny	mild	high	N
3	overcast	hot	high	Y
4	rain	mild	high	Y
5	sunny	cool	normal	Y
6	rain	cool	normal	N
7	overcast	cool	normal	Y
8	sunny	mild	high	?

$$\begin{aligned}
(PG = Y|i_8) &= \frac{P(i_8|PG = Y)P(PG = Y)}{P(i_8)} \\
&= P(O = Sunny, T = mild, H = high|PG = Y) \\
&\quad \times \frac{P(PG = Y)}{P(i_8)} \\
&= P(O = Sunny|PG = Y) \times P(T = mild|PG = Y) \\
&\quad \times P(H = high|PG = Y) \times \frac{P(PG = Y)}{P(i_8)} \\
&= \frac{1}{4} \times \frac{1}{4} \times \frac{2}{4} \times \frac{\frac{4}{7}}{P(i_8)} = \frac{1}{56P(i_8)}.
\end{aligned}$$

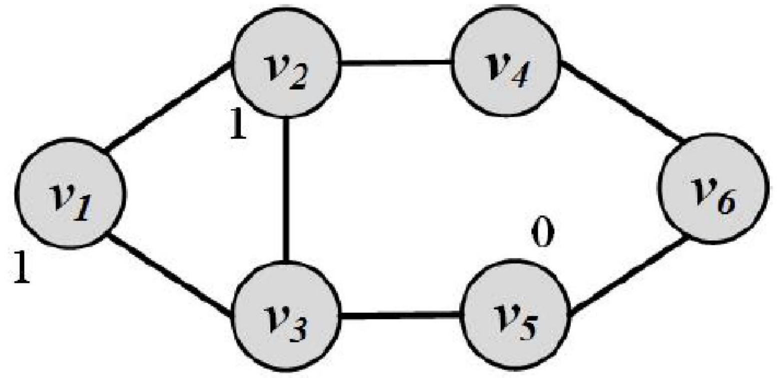
$$\begin{aligned}
P(PG = N|i_8) &= \frac{P(i_8|PG = N)P(PG = N)}{P(i_8)} \\
&= P(O = Sunny, T = mild, H = high|PG = N) \\
&\quad \times \frac{P(PG = N)}{P(i_8)} \\
&= P(O = Sunny|PG = N) \times P(T = mild|PG = N) \\
&\quad \times P(H = high|PG = N) \times \frac{P(PG = N)}{P(i_8)} \\
&= \frac{2}{3} \times \frac{1}{3} \times \frac{2}{3} \times \frac{\frac{3}{7}}{P(i_8)} = \frac{4}{63P(i_8)}.
\end{aligned}$$

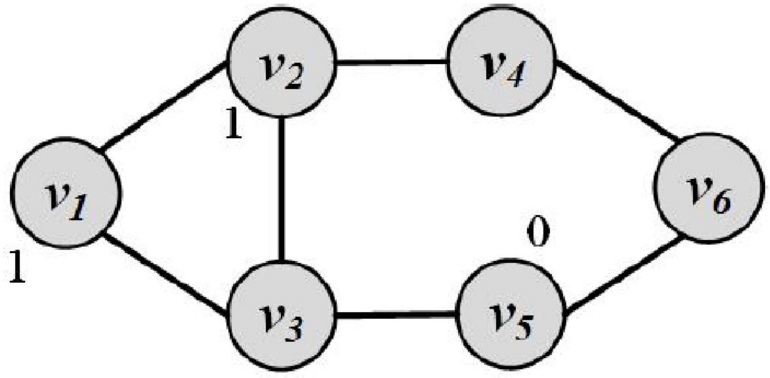
$$P(X|y_i) = \Pi_{j=1}^{n} P(x_j|y_i)$$

$$\frac{1}{56P(i_8)} < \frac{4}{63P(i_8)} \dashrightarrow Play\ Golf = N$$

Classification with Network Information

- Consider a friendship network on social media and a product being marketed to this network.
- Assume we are given the network with the list of individuals that decided to buy or not buy a product. Our goal is to predict the decision for the undecided individuals.





- Let y_i denote the label for node i. We can assume that:

$$P(y_i = 1) \approx P(y_i = 1 | N(v_i))$$

Weighted-vote Relational-Neighbor (wvRN)

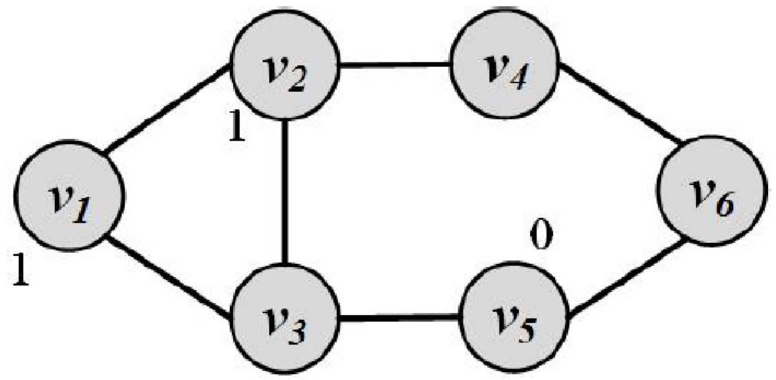
- To find the label of a node, we can perform a weighted vote among its neighbors

$$P(y_i = 1|N(v_i)) = \frac{1}{|N(v_i)|} \sum_{v_j \in N(v_i)} P(y_j = 1|N(v_j)).$$

Note that $P(y_i = 1|N(v_i))$ is *only* calculated for v_i's that are unlabeled.

- Note that we need to compute these probabilities using some order until convergence [i.e., they don't change]

wvRN example



$$P(y_1 = 1|N(v_1)) = 1,$$
$$P(y_2 = 1|N(v_2)) = 1,$$
$$P(y_5 = 1|N(v_5)) = 0.$$

$$\begin{aligned} P(y_3|N(v_3)) & \\ &= \frac{1}{|N(v_3)|}\sum_{v_j \in N(v_3)} P(y_j = 1|N(v_j)) \\ &= \frac{1}{3}(P(y_1 = 1|N(v_1)) + P(y_2 = 1|N(v_2)) + P(y_5 = 1|N(v_5))) \\ &= \frac{1}{3}(1 + 1 + 0) = 0.67. \end{aligned}$$

$$P(y_4|N(v_4)) = \frac{1}{2}(1 + 0.5) = 0.75,$$
$$P(y_6|N(v_6)) = \frac{1}{2}(0.75 + 0) = 0.38.$$

$$P_{(1)}(y_4|N(v_4)) = \frac{1}{2}(1 + 0.38) = 0.69,$$
$$P_{(1)}(y_6|N(v_6)) = \frac{1}{2}(0.69 + 0) = 0.35,$$
$$P_{(2)}(y_4|N(v_4)) = \frac{1}{2}(1 + 0.35) = 0.68,$$
$$P_{(2)}(y_6|N(v_6)) = \frac{1}{2}(0.68 + 0) = 0.34,$$
$$P_{(3)}(y_4|N(v_4)) = \frac{1}{2}(1 + 0.34) = 0.67,$$
$$P_{(3)}(y_6|N(v_6)) = \frac{1}{2}(0.67 + 0) = 0.34,$$
$$P_{(4)}(y_4|N(v_4)) = \frac{1}{2}(1 + 0.34) = 0.67,$$
$$P_{(4)}(y_6|N(v_6)) = \frac{1}{2}(0.67 + 0) = 0.34.$$