

Unit-6

Classification and Prediction

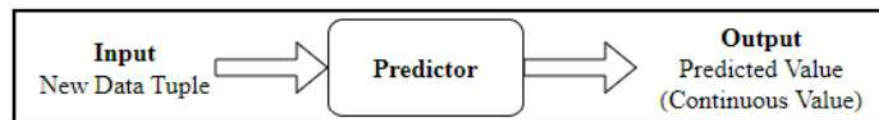
Introduction

Classification and prediction are two forms of data analysis that can be used to extract models describing important data classes or to predict future data trends. Classification models predict categorical (discrete, unordered) class labels; and prediction models predict continuous valued functions.

- **Classification:** Classification is the process of identifying the category or class label of the new observation to which it belongs. For this, Classification extracts the classification model (classifier) from the given training set and then uses this model to decide the category on new data tuple to be classified.



- **Prediction:** Prediction is the process of identifying the missing or unavailable numerical data for a new observation. For this, Prediction learns the continuous valued function from the given training set and then uses this function to predict the continuous value of new data tuple for which prediction is required.

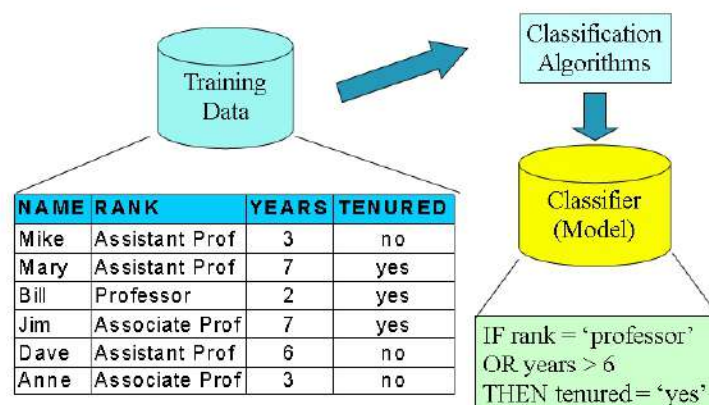


For example, we can build a classification model to categorize bank loan applications as either safe or risky, or a prediction model to predict the expenditures in dollars of potential customers on computer equipment given their income and occupation.

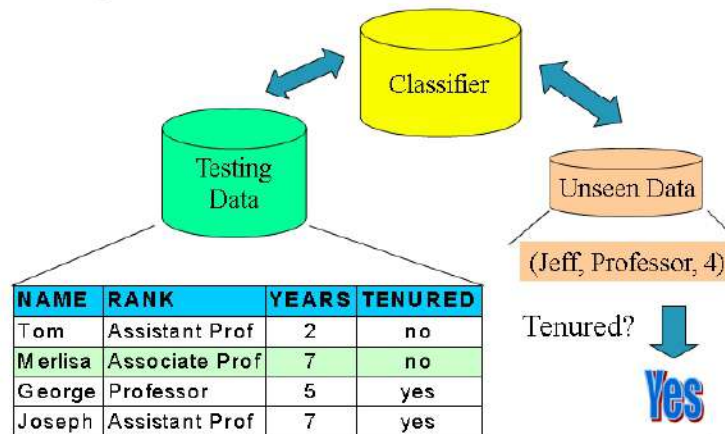
Learning and Testing of Classification

Data classification is a two-step process:

1. **Model or Classifier Construction:** This step is also called training phase or learning stage. In this step a model or classifier is constructed by using any one classification algorithm based on set of training data (Each data object has predefined class label). *E.g.* From given training data set classification algorithm learns model or rule <IF rank = “professor” OR years>6 THEN tenured = “yes”> as shown in figure below:



2. **Model Usage:** In this step, the classifier is used for classification. Here the test dataset is used to estimate the accuracy of classifier. This dataset contains values of input attributes along with class label of the output attribute. Then, accuracy of the classifier is computed by looking at predicted class labels and actual class labels of test dataset. The classifier can be applied to the new data tuples (whose class labels is not known) if the accuracy is considered acceptable. *E.g.*



Classification by Decision Tree Induction

Decision tree induction is the learning of decision trees from class-labeled training tuples. A decision tree is a flowchart-like tree structure, where each internal node (non-leaf node) denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (or terminal node) holds a class label. The top most node in a tree is the root node.

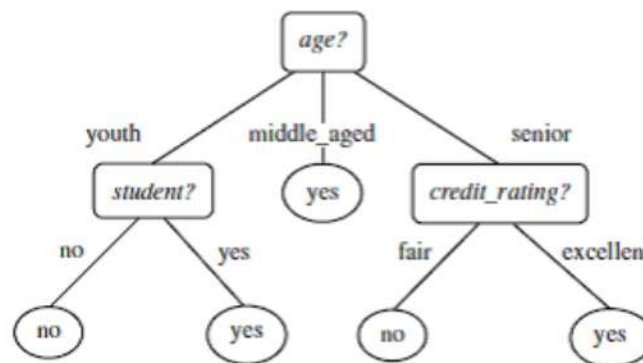


Fig: A decision tree for the concept *buys computer*

Class-label Yes: The customer is likely to buy a computer

Class-label No: The customer is unlikely to buy a computer

Why decision tree classifier are so popular?

- The construction of a decision tree doesn't require any domain knowledge or parameter setting.
- They can handle high dimensional data.
- The learning and classification steps of decision tree are simple and fast.
- They have a high accuracy.
- It is easily understandable by humans.

How are decision trees used for classification?

Given a tuple, X , for which the associated class label is unknown, the attribute values of the tuple are tested against the decision tree. A path is traced from the root to a leaf node, which holds the class prediction for that tuple. Decision trees can easily be converted to classification rules.

Algorithm for Constructing Decision Tree

1. At start, all the training tuples are at the root.
2. Tuples are partitioned recursively based on selected attributes.
3. If all samples for a given node belong to the same class
 - Label the class
4. If There are no remaining attributes for further partitioning
 - **Majority voting** is employed for classifying the leaf
5. There are no samples left
 - Label the class and terminate
6. Else
 - Got to step 2

ID3 as Attribute Selection Measure

- ✓ **ID3 (Iterative Dichotomiser 3)** uses *information gain* as its attribute selection measure.
- ✓ The attribute with the highest information gain is chosen as the splitting attribute for node N .

Information gain is calculated as:

Suppose S is a set of instances, A is an attribute, S_v is the subset of S with $A = v$, and $Values(A)$ is the set of all possible values of A , then

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

Where,

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

Where p_i is the probability that an arbitrary tuple in S belongs to class C_i , estimated by

$$p_i = \frac{|C_{i,S}|}{|S|}$$

Where $|C_{i,S}|$ is the number of tuples in S belonging to class C_i and $|S|$ is the number of tuples in S .

Examples

Q. Consider the following 14 days weather dataset with attributes outlook, temperature, humidity, and wind. The outcome variable will be playing ball or not. Construct decision tree from given dataset using ID3 algorithm and use the constructed decision tree to classify the new data $X = \{\text{Outlook} = \text{Sunny}, \text{Temperature} = \text{Cool}, \text{Humidity} = \text{Normal}, \text{Wind} = \text{Strong}\}$.

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

Solution:

From the total of 14 rows in our dataset S , there are 9 rows with the target value “Yes” and 5 rows with the target value “No”. The entropy of S is calculated as:

$$\text{Entropy}(S) = -\frac{9}{14} \log_2 \frac{9}{14} - \frac{5}{14} \log_2 \frac{5}{14} = 0.94$$

We now calculate the Information Gain for each attribute:

Attribute: Outlook

$$\text{Entropy}(S_{\text{Sunny}}) = -\frac{2}{5} \log_2 \frac{2}{5} - \frac{3}{5} \log_2 \frac{3}{5} = 0.971$$

$$\text{Entropy}(S_{\text{Overcast}}) = -\frac{4}{4} \log_2 \frac{4}{4} - \frac{0}{4} \log_2 \frac{0}{4} = 0$$

$$\text{Entropy}(S_{\text{Rain}}) = -\frac{3}{5} \log_2 \frac{3}{5} - \frac{2}{5} \log_2 \frac{2}{5} = 0.971$$

$$\text{Gain}(S, \text{Outlook}) = 0.94 - \left(\frac{5}{14}\right)(0.971) - \left(\frac{4}{14}\right)(0) - \left(\frac{5}{14}\right)(0.971) = 0.2464$$

Attribute: Temperature

$$\text{Entropy}(S_{\text{Hot}}) = -\frac{2}{4} \log_2 \frac{2}{4} - \frac{2}{4} \log_2 \frac{2}{4} = 1.0$$

$$\text{Entropy}(S_{\text{Mild}}) = -\frac{4}{6} \log_2 \frac{4}{6} - \frac{2}{6} \log_2 \frac{2}{6} = 0.9183$$

$$\text{Entropy}(S_{\text{Cool}}) = -\frac{3}{4} \log_2 \frac{3}{4} - \frac{1}{4} \log_2 \frac{1}{4} = 0.8113$$

$$\text{Gain}(S, \text{Temperature}) = 0.94 - \left(\frac{4}{14}\right)(1.0) - \left(\frac{6}{14}\right)(0.9183) - \left(\frac{4}{14}\right)(0.8113) = 0.0289$$

Attribute: Humidity

$$\text{Entropy}(S_{\text{High}}) = -\frac{3}{7} \log_2 \frac{3}{7} - \frac{4}{7} \log_2 \frac{4}{7} = 0.9852$$

$$\text{Entropy}(S_{\text{Normal}}) = -\frac{6}{7} \log_2 \frac{6}{7} - \frac{1}{7} \log_2 \frac{1}{7} = 0.5916$$

$$\text{Gain}(S, \text{Humidity}) = 0.94 - \left(\frac{7}{14}\right)(0.9852) - \left(\frac{7}{14}\right)(0.5916) = 0.1516$$

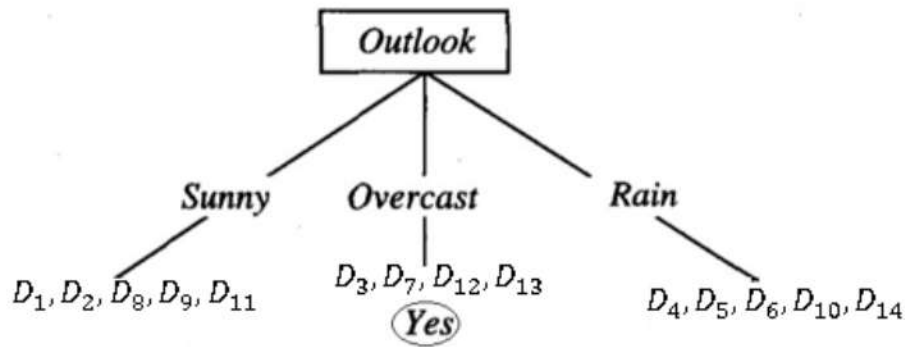
Attribute: Wind

$$\text{Entropy}(S_{\text{Strong}}) = -\frac{3}{6} \log_2 \frac{3}{6} - \frac{3}{6} \log_2 \frac{3}{6} = 1.0$$

$$\text{Entropy}(S_{\text{Weak}}) = -\frac{6}{8} \log_2 \frac{6}{8} - \frac{2}{8} \log_2 \frac{2}{8} = 0.8113$$

$$\text{Gain}(S, \text{Wind}) = 0.94 - \left(\frac{6}{14}\right)(1.0) - \left(\frac{8}{14}\right)(0.8113) = 0.0478$$

The attribute outlook has the **highest information gain** of 0.246, thus it is chosen as root.



Here, when Outlook = Overcast, it is of pure class (Yes). Now, we have to repeat same procedure for the data with rows consist of Outlook value as Sunny and then for Outlook value as Rain.

Now, finding the best attribute for splitting the data with Outlook=Sunny values { Dataset rows = [1, 2, 8, 9, 11]}.

Day	Temperature	Humidity	Wind	Play ball
D1	Hot	High	Weak	No
D2	Hot	High	Strong	No
D8	Mild	High	Weak	No
D9	Cool	Normal	Weak	Yes
D11	Mild	Normal	Strong	Yes

Complete Entropy of “Sunny” is:

$$Entropy(S_{Sunny}) = -\frac{2}{5} \log_2 \frac{2}{5} - \frac{3}{5} \log_2 \frac{3}{5} = 0.971$$

Calculating Information gain for attributes with respect to Sunny is:

Attribute: Temperature

$$Entropy(S_{Hot}) = 0.0$$

$$Entropy(S_{Mild}) = 1.0$$

$$Entropy(S_{Cool}) = 0.0$$

$$Gain(S_{Sunny}, Temperature) = 0.971 - \left(\frac{2}{5}\right)(0) - \left(\frac{2}{5}\right)(1.0) - \left(\frac{1}{5}\right)(0) = 0.571$$

Attribute: Humidity

$$Entropy(S_{High}) = 0.0$$

$$Entropy(S_{Normal}) = 0.0$$

$$Gain(S_{Sunny}, Humidity) = 0.971 - \left(\frac{3}{5}\right)(0) - \left(\frac{2}{5}\right)(0) = 0.971$$

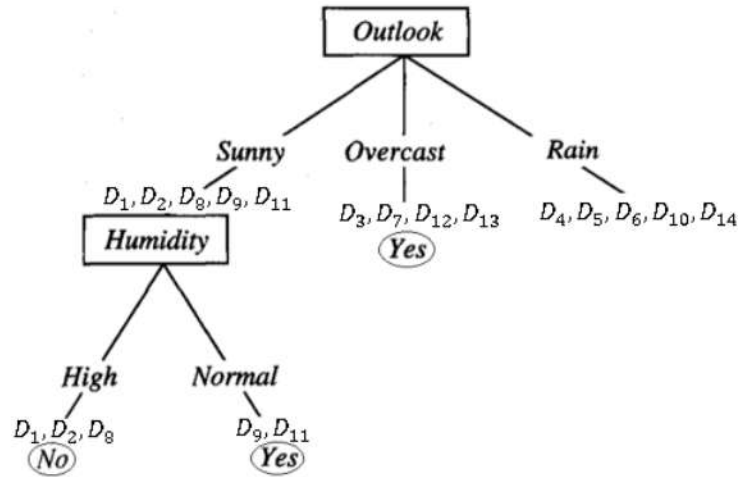
Attribute: Wind

$$Entropy(S_{Strong}) = 1.0$$

$$Entropy(S_{Weak}) = -\frac{1}{3} \log_2 \frac{1}{3} - \frac{2}{3} \log_2 \frac{2}{3} = 0.9183$$

$$Gain(S_{Sunny}, Wind) = 0.971 - \left(\frac{2}{5}\right)(1.0) - \left(\frac{3}{5}\right)(0.9183) = 0.02$$

The information gain for humidity is highest, therefore it is chosen as the next node.



Here, when Outlook = Sunny and Humidity = High, it is a pure class of category "No". And When Outlook = Sunny and Humidity = Normal, it is again a pure class of category "Yes". Therefore, we don't need to do further calculations.

Now, finding the best attribute for splitting the data with Outlook = Rain values { Dataset rows = [4, 5, 6, 10, 14]}.

Day	Temperature	Humidity	Wind	Play ball
D4	Mild	High	Weak	Yes
D5	Cool	Normal	Weak	Yes
D6	Cool	Normal	Strong	No
D10	Mild	Normal	Weak	Yes
D14	Mild	High	Strong	No

Complete Entropy of Rain is:

$$Entropy(S_{Rain}) = -\frac{3}{5} \log_2 \frac{3}{5} - \frac{2}{5} \log_2 \frac{2}{5} = 0.971$$

Calculating Information gain for attributes with respect to Rain is:

Attribute: Temperature

$$Entropy(S_{Hot}) = 0.0$$

$$Entropy(S_{Mild}) = -\frac{2}{3} \log_2 \frac{2}{3} - \frac{1}{3} \log_2 \frac{1}{3} = 0.9183$$

$$Entropy(S_{Cool}) = 1.0$$

$$Gain(S_{Rain}, Temperature) = 0.971 - \left(\frac{0}{5}\right)(0) - \left(\frac{3}{5}\right)(0.9183) - \left(\frac{2}{5}\right)(1.0) = 0.02$$

Attribute: Humidity

$$Entropy(S_{High}) = 1.0$$

$$Entropy(S_{Normal}) = -\frac{2}{3} \log_2 \frac{2}{3} - \frac{1}{3} \log_2 \frac{1}{3} = 0.9183$$

$$Gain(S_{Rain}, Humidity) = 0.971 - \left(\frac{2}{5}\right)(1.0) - \left(\frac{3}{5}\right)(0.9183) = 0.02$$

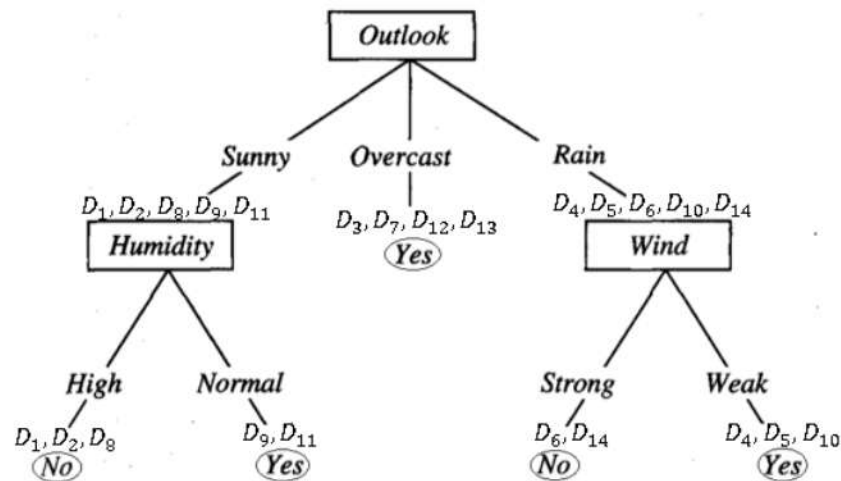
Attribute: Wind

$$Entropy(S_{Strong}) = 0.0$$

$$Entropy(S_{Weak}) = 0.0$$

$$Gain(S_{Rain}, Wind) = 0.971 - \left(\frac{2}{5}\right)(0) - \left(\frac{3}{5}\right)(0) = 0.971$$

Here, the attribute with maximum information gain is Wind. So, the decision tree built so far -



Here, when Outlook = Rain and Wind = Strong, it is a pure class of category "No". And When Outlook = Rain and Wind = Weak, it is again a pure class of category "Yes". And this is our final desired tree for the given dataset.

Now,

Given Test tuple: $X = \{Outlook = Sunny, Temperature = Cool, Humidity = Normal, Wind = Strong\}$

Thus, the predicted class for the given data is: **Play ball = Yes**

Q. Consider the following 14 training dataset assumed a credit risk of high, moderate or low to people based on the following properties of their credit rating:

- Credit history with possible values {Good, Bad, Unknown}
- Debt with possible values {High, Low}
- Collateral with possible values {Adequate, None}
- Income with possible values {"\$0 to \$15", "\$15 to \$35", "Over \$35"}

No.	Credit history	Debt	Collateral	Income	Credit Risk
1	bad	high	none	\$0 to \$15k	high
2	unknown	high	none	\$15 to \$35k	high
3	unknown	low	none	\$15 to \$35k	moderate
4	unknown	low	none	\$0 to \$15k	high
5	unknown	low	none	over \$35k	low
6	unknown	low	adequate	over \$35k	low
7	bad	low	none	\$0 to \$15k	high
8	bad	low	adequate	over \$35k	moderate
9	good	low	none	over \$35k	low
10	good	high	adequate	over \$35k	low
11	good	high	none	\$0 to \$15k	high
12	good	high	none	\$15 to \$35k	moderate
13	good	high	none	over \$35k	low
14	bad	high	none	\$15 to \$35k	high

Classify the individual with Credit history = unknown, Debt = low, Collateral = adequate and Income = \$15 to \$35 using decision tree algorithm. Use ID3 algorithm for building the decision tree.

Solution:

From the total of 14 rows in our dataset S , there are 6 rows with the target value “high”, 6 rows with the target value “moderate” and 5 rows with the target value “low”. The entropy of S is calculated as:

$$\text{Entropy}(S) = -\frac{6}{14} \log_2 \frac{6}{14} - \frac{6}{14} \log_2 \frac{6}{14} - \frac{5}{14} \log_2 \frac{5}{14} = 1.531$$

We now calculate the Information Gain for each attribute:

Attribute: Credit History

$$\text{Entropy}(S_{bad}) = -\frac{3}{4} \log_2 \frac{3}{4} - \frac{1}{4} \log_2 \frac{1}{4} - \frac{0}{4} \log_2 \frac{0}{4} = 0.811$$

$$\text{Entropy}(S_{unknown}) = -\frac{2}{5} \log_2 \frac{2}{5} - \frac{1}{5} \log_2 \frac{1}{5} - \frac{1}{5} \log_2 \frac{1}{5} = 1.457$$

$$\text{Entropy}(S_{good}) = -\frac{1}{5} \log_2 \frac{1}{5} - \frac{1}{5} \log_2 \frac{1}{5} - \frac{3}{5} \log_2 \frac{3}{5} = 1.371$$

$$\text{Gain}(S, \text{Credit History}) = 1.531 - \left(\frac{4}{14}\right)(0.811) - \left(\frac{5}{14}\right)(1.457) - \left(\frac{5}{14}\right)(1.371) = 0.289$$

Attribute: Debt

$$\text{Entropy}(S_{high}) = -\frac{4}{7} \log_2 \frac{4}{7} - \frac{1}{7} \log_2 \frac{1}{7} - \frac{2}{7} \log_2 \frac{2}{7} = 1.379$$

$$\text{Entropy}(S_{low}) = -\frac{2}{7} \log_2 \frac{2}{7} - \frac{2}{7} \log_2 \frac{2}{7} - \frac{3}{7} \log_2 \frac{3}{7} = 1.557$$

$$\text{Gain}(S, \text{Debt}) = 1.531 - \left(\frac{7}{14}\right)(1.379) - \left(\frac{7}{14}\right)(1.557) = 0.063$$

Attribute: Collateral

$$\text{Entropy}(S_{none}) = -\frac{6}{11} \log_2 \frac{6}{11} - \frac{2}{11} \log_2 \frac{2}{11} - \frac{3}{11} \log_2 \frac{3}{11} = 1.435$$

$$\text{Entropy}(S_{adequate}) = -\frac{0}{3} \log_2 \frac{0}{3} - \frac{1}{3} \log_2 \frac{1}{3} - \frac{2}{3} \log_2 \frac{2}{3} = 0.918$$

$$\text{Gain}(S, \text{Collateral}) = 1.531 - \left(\frac{11}{14}\right)(1.435) - \left(\frac{3}{14}\right)(0.918) = 0.208$$

Attribute: Income

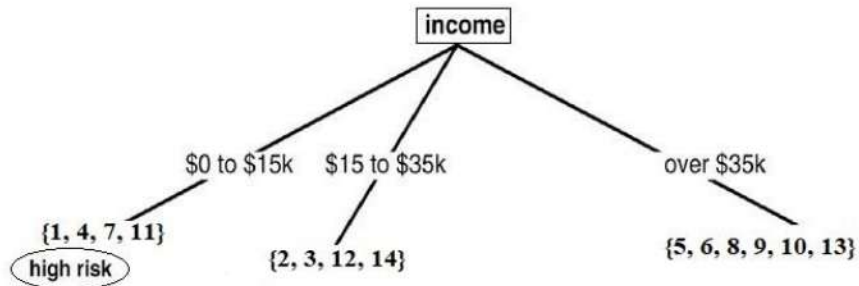
$$\text{Entropy}(S_{\$0 \text{ to } \$15}) = -\frac{4}{4} \log_2 \frac{4}{4} - \frac{0}{4} \log_2 \frac{0}{4} - \frac{0}{4} \log_2 \frac{0}{4} = 0$$

$$\text{Entropy}(S_{\$15 \text{ to } \$35}) = -\frac{2}{4} \log_2 \frac{2}{4} - \frac{2}{4} \log_2 \frac{2}{4} - \frac{0}{4} \log_2 \frac{0}{4} = 1$$

$$\text{Entropy}(S_{\text{over } \$35}) = -\frac{0}{6} \log_2 \frac{0}{6} - \frac{1}{6} \log_2 \frac{1}{6} - \frac{5}{6} \log_2 \frac{5}{6} = 0.650$$

$$\text{Gain}(S, \text{Income}) = 1.531 - \left(\frac{4}{14}\right)(0) - \left(\frac{4}{14}\right)(1) - \left(\frac{6}{14}\right)(0.650) = 0.967$$

The attribute Income has the **highest information gain** of 0.967, thus it is chosen as root.



Here, when Income = \$0 to \$15k, it is of pure class (high risk). Now, we have to repeat same procedure for the data with rows consist of Income value as \$15 to \$35k and then for Income value as over \$35k.

Now, finding the best attribute for splitting the data with Income = \$15 to \$35k values {Dataset rows = [2, 3, 12, 14]}.

No.	Credit History	Debt	Collateral	Credit Risk
2	unknown	high	none	high
3	unknown	low	none	moderate
12	good	high	none	moderate
14	bad	high	none	high

Complete entropy of "\$15 to \$35k" is:

$$Entropy(S_{\$15\ to\ \$35k}) = -\frac{2}{4} \log_2 \frac{2}{4} - \frac{2}{4} \log_2 \frac{2}{4} = 1$$

Next, we calculate the Information Gain for attributes with respect to \$15 to \$35k is:

Attribute: Credit History

$$Entropy(S_{bad}) = -\frac{1}{1} \log_2 \frac{1}{1} - \frac{0}{1} \log_2 \frac{0}{1} = 0$$

$$Entropy(S_{unknown}) = -\frac{1}{2} \log_2 \frac{1}{2} - \frac{1}{2} \log_2 \frac{1}{2} = 1$$

$$Entropy(S_{good}) = -\frac{1}{1} \log_2 \frac{1}{1} - \frac{0}{1} \log_2 \frac{0}{1} = 0$$

$$Gain(S_{\$15\ to\ \$35k}, Credit\ History) = 1 - \left(\frac{1}{4}\right)(0) - \left(\frac{2}{4}\right)(1) - \left(\frac{1}{4}\right)(0) = 0.5$$

Attribute: Debt

$$Entropy(S_{high}) = -\frac{2}{3} \log_2 \frac{2}{3} - \frac{1}{3} \log_2 \frac{1}{3} = 0.918$$

$$Entropy(S_{low}) = -\frac{1}{1} \log_2 \frac{1}{1} - \frac{0}{1} \log_2 \frac{0}{1} = 0$$

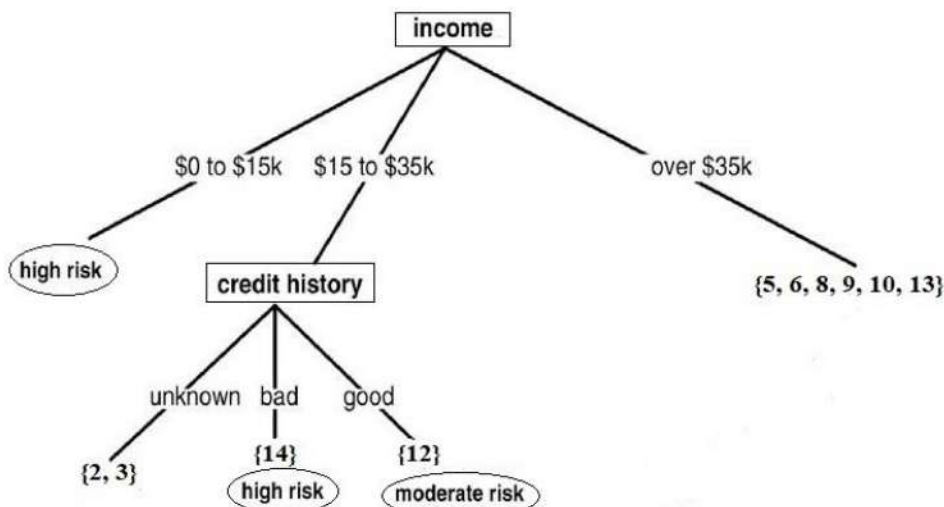
$$Gain(S_{\$15\ to\ \$35k}, Debt) = 1 - \left(\frac{3}{4}\right)(0.918) - \left(\frac{1}{4}\right)(0) = 0.31$$

Attribute: Collateral

$$Entropy(S_{none}) = -\frac{2}{4} \log_2 \frac{2}{4} - \frac{2}{4} \log_2 \frac{2}{4} = 1$$

$$Gain(S_{\$15\ to\ \$35k}, Collateral) = 1 - \left(\frac{4}{4}\right)(1) = 0$$

Here, the attribute with maximum information gain is Credit History. So, the decision tree built so far -



Here, when income = \$15 to \$35k and credit history = bad, it is a pure class of category "high risk". And when income = \$15 to \$35k and credit history = good, it is again a pure class of category "moderate risk".

Again, finding the best attribute for splitting the data with income = \$15 to \$35k and credit history = unknown values { Dataset rows = [2, 3]}.

No.	Debt	Collateral	Credit Risk
2	high	none	high
3	low	none	moderate

Complete Entropy for income = \$15 to \$35k and credit history = unknown:

$$Entropy(S_{unknown}) = -\frac{1}{2} \log_2 \frac{1}{2} - \frac{1}{2} \log_2 \frac{1}{2} = 1$$

Calculating Information Gain for attributes Debt and Collateral:

Attribute: Debt

$$Entropy(S_{high}) = -\frac{1}{1} \log_2 \frac{1}{1} - \frac{0}{1} \log_2 \frac{0}{1} = 0$$

$$Entropy(S_{low}) = -\frac{1}{1} \log_2 \frac{1}{1} - \frac{0}{1} \log_2 \frac{0}{1} = 0$$

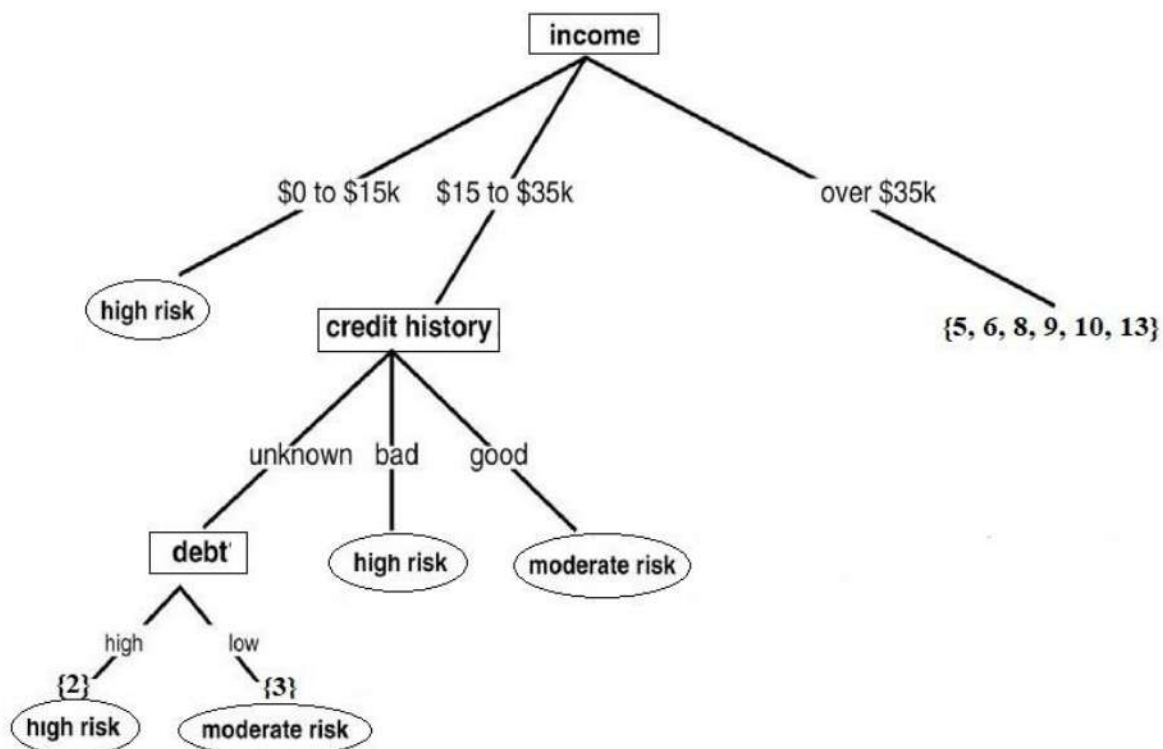
$$Gain(S_{unknown}, Debt) = 1 - \left(\frac{1}{2}\right)(0) - \left(\frac{1}{2}\right)(0) = 1$$

Attribute: Collateral

$$Entropy(S_{none}) = -\frac{1}{2} \log_2 \frac{1}{2} - \frac{1}{2} \log_2 \frac{1}{2} = 1$$

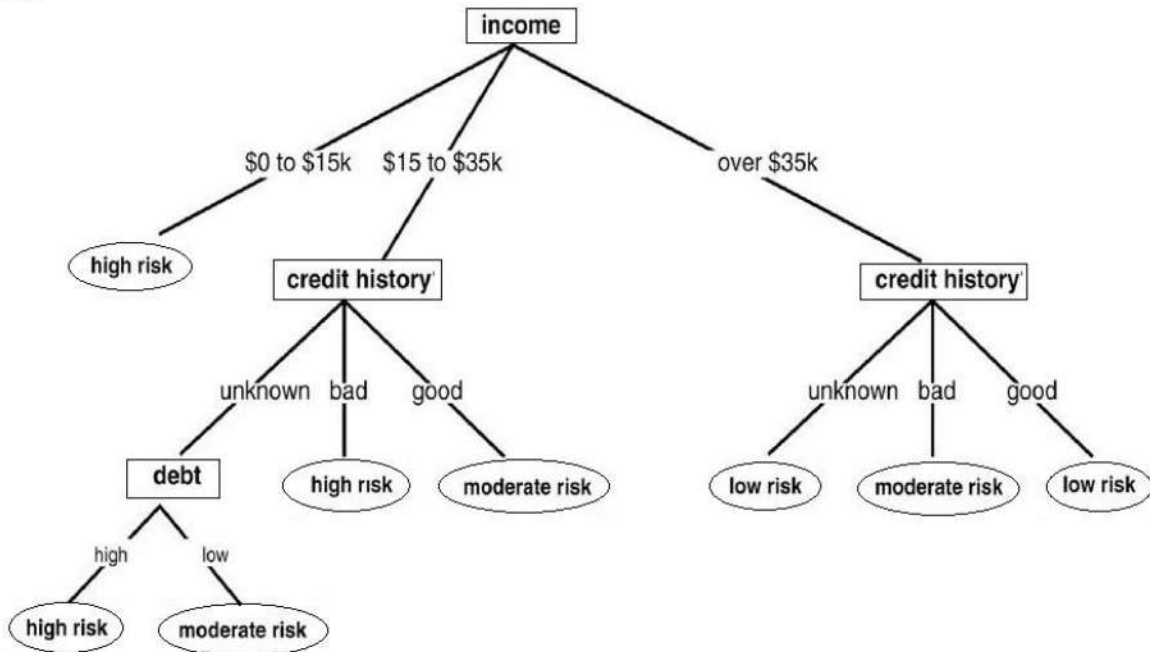
$$Gain(S_{unknown}, Collateral) = 1 - \left(\frac{2}{2}\right)(1) = 0$$

The information gain for Debt is highest, therefore it is chosen as the next node.



Here, when income = \$15 to \$35k, credit history = unknown and debt = high, it is a pure class of category "high risk". And when income = \$15 to \$35k, credit history = unknown and debt = low, it is again a pure class of category "moderate risk".

The following tree will be obtained by applying the same procedure as above for *income "above \$35k"*



Given Test tuple: $X = \{Credit\ history = unknown, Debt = low, Collateral = adequate\ and\ Income = \$15\ to\ \$35\}$

Thus, the predicted class for the given data is: **Credit Risk = moderate risk**

Bayesian Classification

Bayesian classifiers are statistical classifiers. They can predict class membership probabilities, such as the probability that a given tuple belongs to a particular class. Bayesian classification is based on Bayes’ theorem. It is also called *Naïve Bayes Classification* or *Naïve Bayesian Classification*.

Suppose that there are m classes, C_1, C_2, \dots, C_m . Given a tuple, $X = (x_1, x_2, \dots, x_n)$, the classifier will predict that X belongs to the class having the highest posterior probability, conditioned on X . That is, the naïve Bayesian classifier predicts that tuple X belongs to the class C_i if and only if

$$P(C_i|X) > P(C_j|X) \text{ for } 1 \leq j \leq m, j \neq i.$$

By Bayes’ theorem,

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

Now, as the denominator remains constant for a given input, we can remove that term:

$$P(C_i|X) = P(X|C_i)P(C_i)$$

The probability $P(X|C_i)$ is given by the equation given below:

$$P(X|C_i) = \prod_{k=1}^n P(x_k|C_i) = P(x_1|C_i) * P(x_2|C_i) * \dots * P(x_n|C_i)$$

Examples**Q. Consider the following data set.**

Confident	Studied	Sick	Result
Yes	No	No	Fail
Yes	No	Yes	Pass
No	Yes	Yes	Fail
No	Yes	No	Pass
Yes	Yes	Yes	Pass

Find out whether the object with attribute Confident = Yes, Studied = Yes and Sick = No will Fail or Pass using Bayesian classification.

Solution:

Let $X = (\text{Confident} = \text{Yes}, \text{Studied} = \text{Yes}, \text{Sick} = \text{No})$

First we need to calculate the class probabilities i.e.

$$P(\text{Result} = \text{Pass}) = \frac{3}{5} = 0.6$$

$$P(\text{Result} = \text{Fail}) = \frac{2}{5} = 0.4$$

Now we compute the following conditional probabilities:

$$P(\text{Confident} = \text{Yes} | \text{Result} = \text{Pass}) = \frac{2}{3}$$

$$P(\text{Studied} = \text{Yes} | \text{Result} = \text{Pass}) = \frac{2}{3}$$

$$P(\text{Sick} = \text{No} | \text{Result} = \text{Pass}) = \frac{1}{3}$$

$$P(\text{Confident} = \text{Yes} | \text{Result} = \text{Fail}) = \frac{1}{2}$$

$$P(\text{Studied} = \text{Yes} | \text{Result} = \text{Fail}) = \frac{1}{2}$$

$$P(\text{Sick} = \text{No} | \text{Result} = \text{Fail}) = \frac{1}{2}$$

Using the above probabilities, we obtain:

$$P(X | \text{Result} = \text{Pass}) = \frac{2}{3} * \frac{2}{3} * \frac{1}{3} = 0.148$$

$$P(X | \text{Result} = \text{Fail}) = \frac{1}{2} * \frac{1}{2} * \frac{1}{2} = 0.125$$

Finally,

$$P(\text{Result} = \text{Pass} | X) = P(X | \text{Result} = \text{Pass}) * P(\text{Result} = \text{Pass}) = 0.148 * 0.6 = 0.089$$

$$P(\text{Result} = \text{Fail} | X) = P(X | \text{Result} = \text{Fail}) * P(\text{Result} = \text{Fail}) = 0.125 * 0.4 = 0.05$$

As $0.089 > 0.05$, the instance with $(\text{Confident} = \text{Yes}, \text{Studied} = \text{Yes}, \text{Sick} = \text{No})$ has result as '**Pass**'.

Q. Consider the following dataset.

age	income	student	credit_rating	Class: buys_computer
youth	high	no	fair	no
youth	high	no	excellent	no
middle_aged	high	no	fair	yes
senior	medium	no	fair	yes
senior	low	yes	fair	yes
senior	low	yes	excellent	no
middle_aged	low	yes	excellent	yes
youth	medium	no	fair	no
youth	low	yes	fair	yes
senior	medium	yes	fair	yes
youth	medium	yes	excellent	yes
middle_aged	medium	no	excellent	yes
middle_aged	high	yes	fair	yes
senior	medium	no	excellent	no

Predict class level of the tuple: $X = (\text{age} = \text{youth}, \text{income} = \text{medium}, \text{student} = \text{yes}, \text{credit_rating} = \text{fair})$ using Bayesian classification.

Solution:

First we need to calculate the class probabilities i.e.

$$P(\text{buys_computer} = \text{yes}) = \frac{9}{14} = 0.643$$

$$P(\text{buys_computer} = \text{no}) = \frac{5}{14} = 0.357$$

Now we compute the following conditional probabilities:

$$P(\text{age} = \text{youth} | \text{buys_computer} = \text{yes}) = 2/9 = 0.222$$

$$P(\text{income} = \text{medium} | \text{buys_computer} = \text{yes}) = 4/9 = 0.444$$

$$P(\text{student} = \text{yes} | \text{buys_computer} = \text{yes}) = 6/9 = 0.667$$

$$P(\text{credit_rating} = \text{fair} | \text{buys_computer} = \text{yes}) = 6/9 = 0.667$$

$$P(\text{age} = \text{youth} | \text{buys_computer} = \text{no}) = 3/5 = 0.6$$

$$P(\text{income} = \text{medium} | \text{buys_computer} = \text{no}) = 2/5 = 0.4$$

$$P(\text{student} = \text{yes} | \text{buys_computer} = \text{no}) = 1/5 = 0.2$$

$$P(\text{credit_rating} = \text{fair} | \text{buys_computer} = \text{no}) = 2/5 = 0.4$$

Using the above probabilities, we obtain:

$$P(X | \text{buys_computer} = \text{yes}) = 0.222 * 0.444 * 0.667 * 0.667 = 0.044$$

$$P(X | \text{buys_computer} = \text{no}) = 0.6 * 0.4 * 0.2 * 0.4 = 0.019$$

Finally,

$$\begin{aligned} P(\text{buys_computer} = \text{yes} | X) &= P(X | \text{buys_computer} = \text{yes}) * P(\text{buys_computer} = \text{yes}) \\ &= 0.044 * 0.643 = 0.028 \end{aligned}$$

$$\begin{aligned} P(\text{buys_computer} = \text{no} | X) &= P(X | \text{buys_computer} = \text{no}) * P(\text{buys_computer} = \text{no}) \\ &= 0.019 * 0.357 = 0.007 \end{aligned}$$

Since $P(\text{buys_computer} = \text{yes} | X) > P(\text{buys_computer} = \text{no} | X)$, Prediction for X is:

$\text{buys_computer} = \text{yes}$

Laplace Smoothing

We have Bayesian classification formula

$$P(C_i | x_1, x_2, \dots, x_n) = P(x_1 | C_i) * P(x_2 | C_i) * \dots * P(x_n | C_i) * P(C_i)$$

If one or more $P(x_i | C_i)$ is absent in the data:

- The probability becomes zero.
- It makes the whole RHS zero.
- This is called “*the problem of zero probability*”.

For example consider our above example: If, say, there are no training tuples representing students for the class $\text{buys_computer} = \text{no}$, then probability will become 0 and we cannot perform correct classification. So, to overcome such problem we can use **Laplace Smoothing (Laplacian Correction)**.

Laplace smoothing is a smoothing technique that handles the problem of zero probability in Naïve Bayes. It adds 1 to each count. The main philosophy behind the concept is that the database is normally large and adding 1 to the count makes negligible difference in probability.

Example

Suppose that a database contains 1000 tuples. For the class $\text{buys_computer} = \text{yes}$ we have 0 tuples with $\text{income} = \text{low}$, 990 tuples with $\text{income} = \text{medium}$, and 10 tuples with $\text{income} = \text{high}$. The probabilities of these events, without the Laplacian correction, are 0, 0.990 (from 990/1000), and 0.010 (from 10/1000), respectively.

Using the Laplacian correction for the three quantities, we pretend that we have 1 more tuple for each income-value pair. In this way, we obtain the following probabilities (rounded up to three decimal places):

$$\frac{1}{1003} = 0.001,$$

$$\frac{991}{1003} = 0.988,$$

$$\frac{11}{1003} = 0.011 \text{ respectively.}$$

The “corrected” probability estimates are close to their “uncorrected” counterparts, yet the zero probability value is avoided.

Classification by Back Propagation

- Backpropagation is a **neural network** learning algorithm.
- A **neural network**: A set of connected input/output units where each connection has a weight associated with it.
- During the learning phase, the **network learns by adjusting the weights** so as to be able to predict the correct class label of the input tuples.

Back-Propagation Algorithm

Step 1: Randomly choose the initial weights and biases.

Step 2: Calculate the actual output:

- For each unit j in the input layer, its output is equal to its input i.e. $O_j = I_j$
- For each unit j in a hidden or output layer:
 - Net input (I_j) = $\sum_i w_{ij} O_i + \theta_j$
Where, w_{ij} is the weight of the connection from unit i in the previous layer to unit j ; O_i is the output of unit i from the previous layer and θ_j is the bias of the unit j .
 - Output of unit j (O_j) = $\frac{1}{1+e^{-I_j}}$

Step 3: Calculate the error:

- For each unit j in the output layer : $Err_j = O_j(1 - O_j)(T_j - O_j)$
Where, O_j is the actual output of unit j and T_j is the targeted output.
- For each unit j in the hidden layers, from last to the first hidden layer:
 $Err_j = O_j(1 - O_j) \sum_k Err_k w_{jk}$
Where, w_{jk} is the weight of the connection from unit j to unit k in the next higher layer, and Err_k is the error of unit k .

Step 4: Update weights and biases:

Weight update:

$$w_{ij}(new) = w_{ij}(old) + l * Err_j * O_i$$

Bias update:

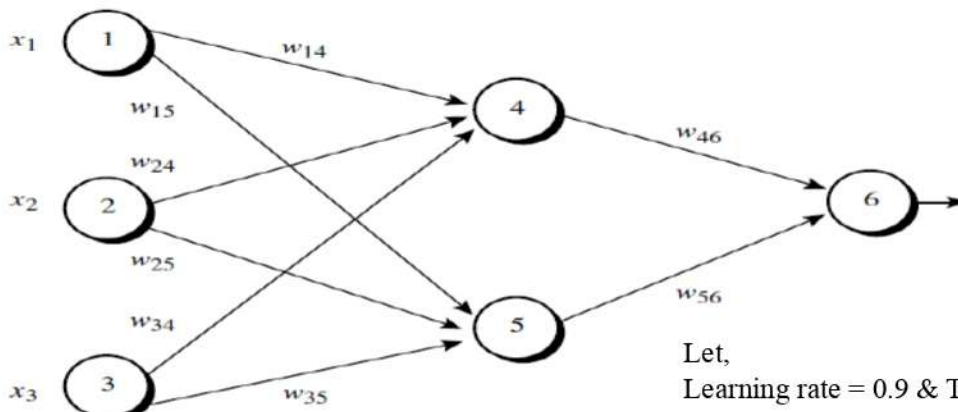
$$\theta_j(new) = \theta_j(old) + l * Err_j$$

Where, l is the learning rate.

Step 5: Repeat step 2 to 5 until the termination condition is met.

Example

Consider a multilayer feed-forward neural network as shown in figure:



Now let us consider the initial input, weight and bias value as:

x_1	x_2	x_3	w_{14}	w_{15}	w_{24}	w_{25}	w_{34}	w_{35}	w_{46}	w_{56}	θ_4	θ_5	θ_6
1	0	1	0.2	-0.3	0.4	0.1	-0.5	0.2	-0.3	-0.2	-0.4	0.2	0.1

Now calculate the output for each node as follows:

For input layer:

$$O_1 = x_1 = 1, O_2 = x_2 = 0, O_3 = x_3 = 1$$

For hidden and output layer:

Node j	Net input, I_j	Output, O_j
4	$0.2 + 0 - 0.5 - 0.4 = -0.7$	$1/(1 + e^{0.7}) = 0.332$
5	$-0.3 + 0 + 0.2 + 0.2 = 0.1$	$1/(1 + e^{-0.1}) = 0.525$
6	$(0.332)(-0.3) + (0.525)(-0.2) + 0.1 = -0.105$	$1/(1 + e^{0.105}) = 0.474$

Then calculate the error at each node:

Node j	Err _j
6	$O_6(1 - O_6)(T - O_6) = 0.474(1 - 0.474)(1 - 0.474) = 0.1311$
5	$O_5(1 - O_5) * Err_6 * w_{56} = 0.525(1 - 0.525) * 0.1311 * (-0.2) = -0.0065$
4	$O_4(1 - O_4) * Err_6 * w_{46} = 0.332(1 - 0.332) * 0.1311 * (-0.3) = -0.0087$

Now update the weight and bias as follows:

$$w_{46}(new) = w_{46}(old) + l * Err_6 * O_4 = -0.3 + 0.9 * 0.1311 * 0.332 = -0.261$$

$$w_{56}(new) = w_{56}(old) + l * Err_6 * O_5 = -0.2 + 0.9 * 0.1311 * 0.525 = -0.138$$

$$w_{14}(new) = w_{14}(old) + l * Err_4 * O_1 = 0.2 + 0.9 * (-0.0087) * 1 = 0.192$$

$$w_{15}(new) = w_{15}(old) + l * Err_5 * O_1 = -0.3 + 0.9 * (-0.0065) * 1 = -0.306$$

$$w_{24}(new) = w_{24}(old) + l * Err_4 * O_2 = 0.4 + 0.9 * (-0.0087) * 0 = 0.4$$

$$w_{25}(new) = w_{25}(old) + l * Err_5 * O_2 = 0.1 + 0.9 * (-0.0065) * 0 = 0.1$$

$$w_{34}(new) = w_{34}(old) + l * Err_4 * O_3 = -0.5 + 0.9 * (-0.0087) * 1 = -0.508$$

$$w_{35}(new) = w_{35}(old) + l * Err_5 * O_3 = 0.2 + 0.9 * (-0.0065) * 1 = 0.194$$

$$\theta_6(new) = \theta_6(old) + l * Err_6 = 0.1 + 0.9 * 0.1311 = 0.218$$

$$\theta_5(new) = \theta_5(old) + l * Err_5 = 0.2 + 0.9 * (-0.0065) = 0.194$$

$$\theta_4(new) = \theta_4(old) + l * Err_4 = -0.4 + 0.9 * (-0.0087) = -0.408$$

Repeat Until $Err_6 \approx 0$

Advantages of Back Propagation Algorithm

- prediction accuracy is generally high
- fast evaluation of the learned target function
- High tolerance to noisy data
- Ability to classify untrained patterns
- Well-suited for continuous-valued inputs and outputs
- Successful on a wide array of real-world data
- Algorithms are inherently parallel

Disadvantages of Back Propagation Algorithm

- long training time
- difficult to understand the learned function (weights)
- Require a number of parameters typically best determined empirically, e.g., the network topology or "structure."
- *Poor interpretability*: Difficult to interpret the symbolic meaning behind the learned weights and of "hidden units" in the network.

Rule Based Classifier

A **rule-based classifier** uses a set of IF-THEN rules for classification. An **IF-THEN** rule is an expression of the form:

$$\text{IF } \textit{condition} \text{ THEN } \textit{conclusion}$$

An example is rule R1,

$$R1: \text{IF } \textit{age} = \textit{youth} \text{ AND } \textit{student} = \textit{'yes'} \text{ THEN } \textit{buys_computer} = \textit{'yes'}$$

R1 can also be written as:

$$R1: (\textit{age} = \textit{youth}) \wedge (\textit{student} = \textit{yes}) \Rightarrow (\textit{buys_computer} = \textit{yes})$$

The “IF” part of a rule is known as the **antecedent** or **precondition**. The “THEN” part is the **consequent**. In the antecedent, the condition consists of one or more *attribute tests* that are logically ANDed. The rule’s consequent contains a class prediction.

If the condition in an **antecedent holds true** for a given tuple, we say that the antecedent is satisfied (or simply, that the rule is satisfied) and that the **rule covers** the tuple.

Rule Coverage & Accuracy

A rule *R* can be assessed by its coverage and accuracy. Given a tuple, *X*, from a class labeled data set, *D*, let *n_{covers}* be the number of tuples that satisfy rules antecedent; *n_{correct}* be the number of tuples correctly classified by *R*; and |*D*| be the number of tuples in *D*. We can define the coverage and accuracy of *R* as below:

- **Rule Coverage:** A rule’s coverage is the percentage of tuples that satisfy rule’s antecedent.

$$\textit{coverage}(R) = \frac{n_{\textit{covers}}}{|D|}$$

- **Rule Accuracy:** A rule’s accuracy is the percentage of tuples that are correctly classified by the rule i.e. satisfy both the antecedent and consequent of a rule.

$$\textit{accuracy}(R) = \frac{n_{\textit{correct}}}{n_{\textit{covers}}}$$

Example

Find coverage and accuracy of the rule

$$R1: (\textit{age} = \textit{youth}) \wedge (\textit{student} = \textit{yes}) \Rightarrow (\textit{buys_computer} = \textit{yes}).$$

Here, size of dataset |*D*| = 14

Number of tuples satisfying the condition (antecedent) of rule (*n_{covers}*) = 2

Therefore, $\textit{coverage}(R1) = \frac{2}{14} = 14.28\%$

It can correctly classify both tuples (i.e. *n_{correct}* = 2)

Therefore, $\textit{accuracy}(R1) = \frac{2}{2} = 100\%$

age	income	student	credit_rating	Class: buys_computer
youth	high	no	fair	no
youth	high	no	excellent	no
middle_aged	high	no	fair	yes
senior	medium	no	fair	yes
senior	low	yes	fair	yes
senior	low	yes	excellent	no
middle_aged	low	yes	excellent	yes
youth	medium	no	fair	no
youth	low	yes	fair	yes
senior	medium	yes	fair	yes
youth	medium	yes	excellent	yes
middle_aged	medium	no	excellent	yes
middle_aged	high	yes	fair	yes
senior	medium	no	excellent	no

Predicting Class Using Rule Based Classification

Example: We would like to classify X according to *buys_computer*:

$$X = (\text{age} = \text{youth}, \text{income} = \text{medium}, \text{student} = \text{yes}, \text{credit_rating} = \text{fair})$$

If a rule is satisfied by X , the rule is said to be **triggered**. X satisfies $R1$, which triggers the rule.

If $R1$ is the only rule satisfied, then the rule **fires** by returning the class prediction for X . If more than one rule is triggered, we need a **conflict resolution strategy** to figure out which rule gets to fire. Two such strategies are: *size ordering* and *rule ordering*.

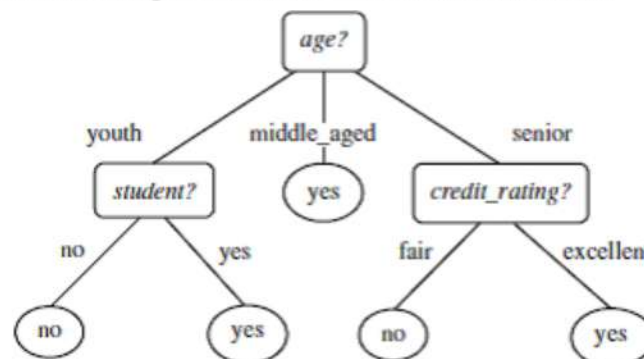
- The **size ordering** scheme fires rule with the most attribute tests in its antecedent.
- The **rule ordering** scheme prioritizes the rules beforehand. The ordering may be *class-based* or *rule-based*.
- With **class-based ordering**, the classes are sorted in order of decreasing importance. That is, all the rules for the most prevalent (or most frequent) class come first, the rules for the next prevalent class come next, and so on.
- With **rule-based ordering**, the rules are organized into one long priority list, according to some measure of rule quality, such as accuracy, coverage, or size, or based on advice from domain experts. When rule ordering is used, the rule set is known as a decision list. With rule ordering, the triggering rule that appears earliest in the list has the highest priority, and so it gets to fire its class prediction. Any other rule that satisfies X is ignored.

If there is no rule satisfied by X then a default rule can be set up to specify a default class, based on a training set. Default class may be the class in majority class of the tuples that were not covered by any rules. The default rule is evaluated at the end, if and only if no other rule covers X . The condition in the default rule is empty.

Rule Extraction from Decision Tree

To extract rules from a decision tree, one rule is created for each path from the root to a leaf node. Each splitting criterion along a given path is logically ANDed to form the rule antecedent (“IF” part). The leaf node holds the class prediction, forming the rule consequent (“THEN” part).

Example: Consider the following decision tree and extract the rules.



$R1$: IF *age* = youth AND *student* = no THEN *buys_computer* = no

$R2$: IF *age* = youth AND *student* = yes THEN *buys_computer* = yes

$R3$: IF *age* = middle_aged THEN *buys_computer* = yes

$R4$: IF *age* = senior AND *credit_rating* = excellent THEN *buys_computer* = yes

$R5$: IF *age* = senior AND *credit_rating* = fair THEN *buys_computer* = no

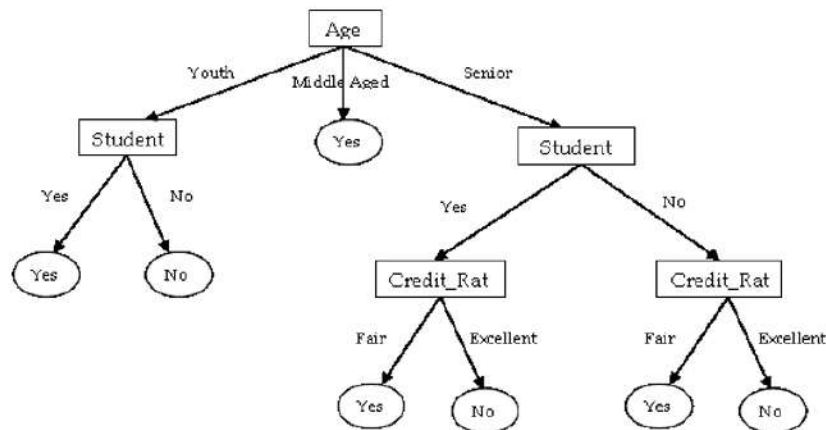
Rules extracted from decision trees are *mutually exclusive* and *exhaustive*.

- **Mutually exclusive rules:** The rules in a rule set R are said to be mutually exclusive if no two rules in R are triggered by the same record. If rules are mutually exclusive then every record is covered by at most one rule.
- **Exhaustive rules:** Rules said to be exhaustive if there is one rule for each possible attribute-value combination. It ensures that every record is covered by at least one rule.

Simplification of rules extracted from decision tree

Rules extracted from decision tree can be simplified. Rules can be simplified by pruning the conditions in the rule antecedent that do not improve the estimated accuracy of the rule.

Example:



Initial Rule:

R: If Age = Senior AND Student = yes AND Credit_rating = Fair THEN buys_computer = yes

Simplified Rule:

R: If Age = Senior AND Credit_Rating = Fair Then Buys_Computer = Yes

Effect of rule simplification is that rules are no longer mutually exclusive and exhaustive.

Advantages and Disadvantages of Rule Based Classifier

Advantages

- As highly expressive as decision trees
- Easy to interpret
- Easy to generate
- Can classify new instances rapidly
- Performance comparable to decision trees

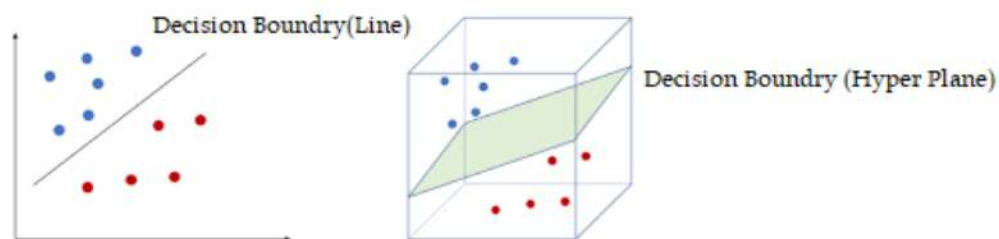
Disadvantages

- If rule set is large then it is complex to apply rule for classification.
- For large training set large number of rules generated these require large amount of memory.
- During rule generation extra computation needed to simplify and prune the rules.

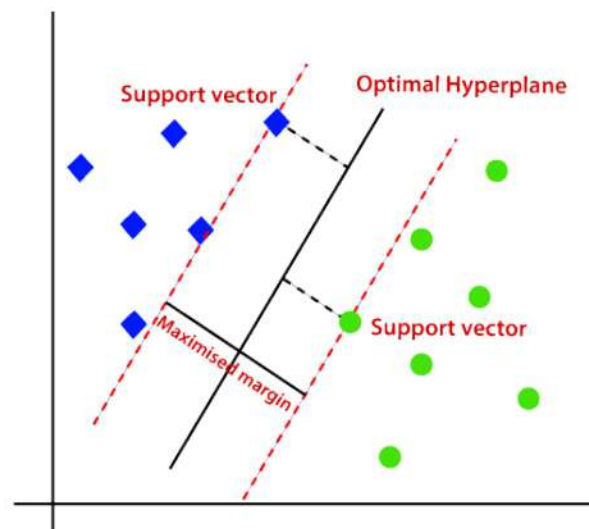
Support Vector Machine

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

A support vector machine takes input data points and outputs the hyperplane (which in two dimensions it's simply a line) that best separates the data points into two classes. This line or hyperplane is the decision boundary: any data points that falls to one side of it is classified in one class and, and the data points that falls to the other of it is classified in another class.

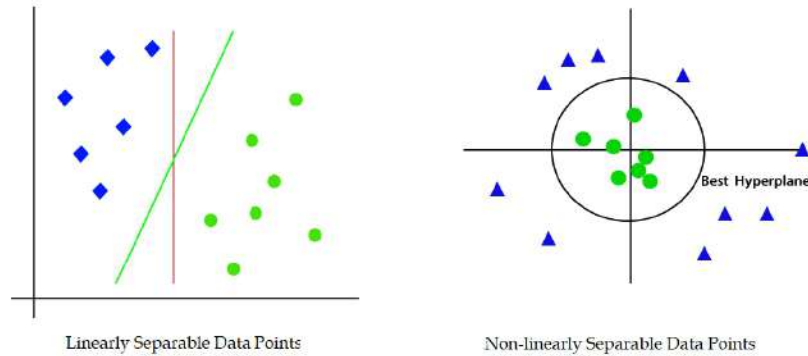


Support vectors are data points that are closest to the hyperplane and influence the position and orientation of the hyperplane. To separate the two classes of data points, there are many possible hyperplanes that could be chosen. SVM algorithm selects optimal hyperplane by choosing hyperplane with largest margin. Such hyperplane is called maximum marginal hyperplane (MMH). Let H_1 and H_2 are planes that passes through support vectors and parallel to the hyperplane of decision boundary. Distance between plane H_1 and the hyperplane should be equal to distance between plane H_2 and the hyperplane. Margin is the distance between planes H_1 and H_2 .



SVM can be of two types:

- **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
- **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.



SVM algorithms use a set of mathematical functions that are defined as the kernel. The SVM kernel is a function that takes low dimensional input space and transforms it into higher-dimensional space, i.e. it converts not separable problem to separable problem. It is mostly useful in non-linear separation problems. Simply put the kernel, it does some extremely complex data transformations then finds out the process to separate the data based on the labels or outputs defined.

Q. Consider following data points:

- **Positively Labelled Data Points:** $(3, 1), (3, -1), (6, 1), (6, -1)$
- **Negatively Labelled Data Points:** $(1, 0), (0, 1), (0, -1), (-1, 0)$

Determine the equation of hyperplane that divides the above data points into two classes.

Solution:

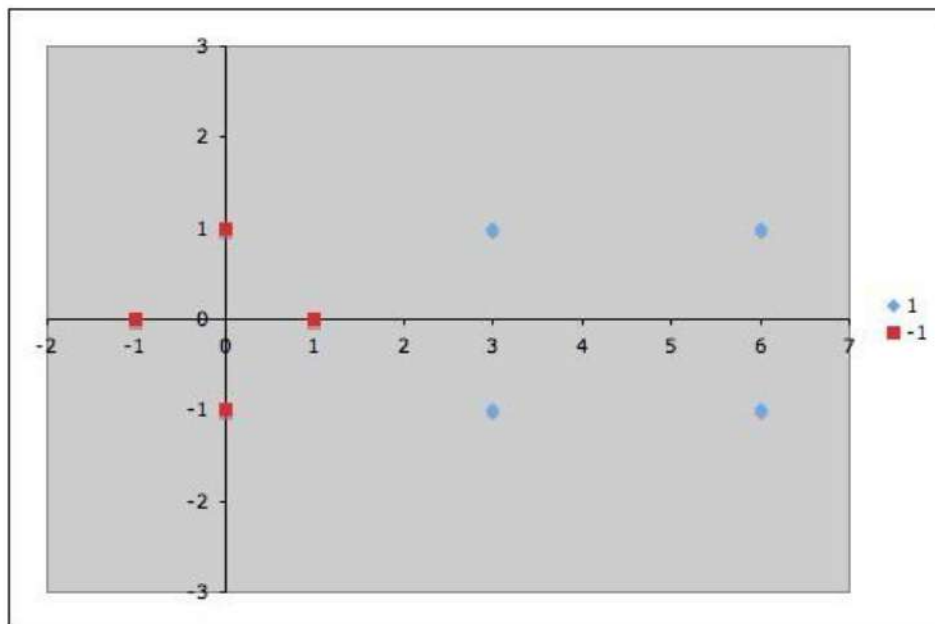


Fig: Sample data points in \mathbb{R}^2 . Blue diamonds are positive example and red squares are negative examples.

Point $(1,0)$ is nearest to negatively labelled data points and $(3,1)$ & $(3,-1)$ are nearest to positively labelled data points. So support vectors are

$$s_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, s_2 = \begin{pmatrix} 3 \\ 1 \end{pmatrix} \text{ and } s_3 = \begin{pmatrix} 3 \\ -1 \end{pmatrix}$$

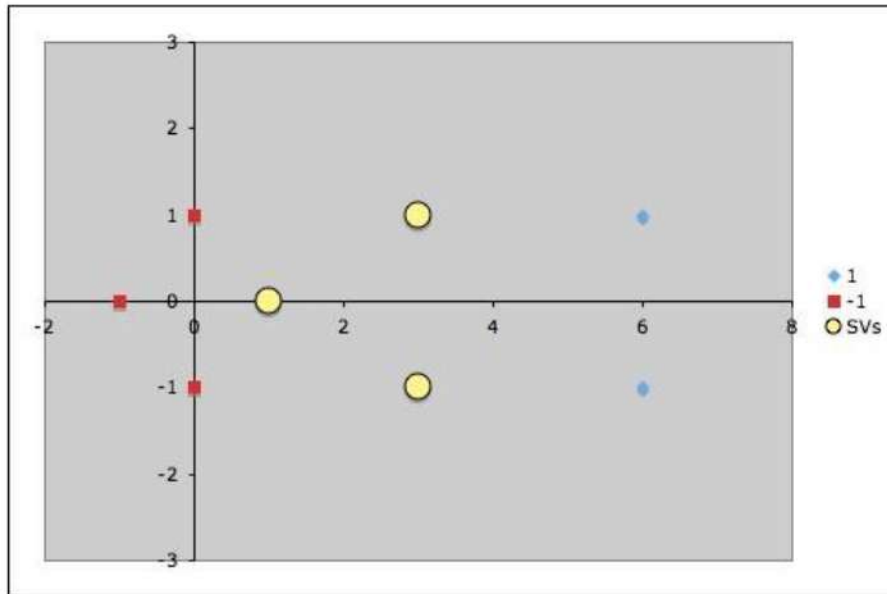


Fig: The three support vectors are marked as yellow circles

Here we will use vectors augmented with a 1 as a bias input, and for clarity we will differentiate these with an over-tilde. That is

$$\tilde{s}_1 = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \tilde{s}_2 = \begin{pmatrix} 3 \\ 1 \\ 1 \end{pmatrix} \text{ and } \tilde{s}_3 = \begin{pmatrix} 3 \\ -1 \\ 1 \end{pmatrix}$$

Now we need to find 3 parameters $\alpha_1, \alpha_2,$ and α_3 based on the following 3 linear equations:

$$\alpha_1 \tilde{s}_1 \cdot \tilde{s}_1 + \alpha_2 \tilde{s}_2 \cdot \tilde{s}_1 + \alpha_3 \tilde{s}_3 \cdot \tilde{s}_1 = -1 \quad (-ve \text{ class})$$

$$\alpha_1 \tilde{s}_1 \cdot \tilde{s}_2 + \alpha_2 \tilde{s}_2 \cdot \tilde{s}_2 + \alpha_3 \tilde{s}_3 \cdot \tilde{s}_2 = +1 \quad (+ve \text{ class})$$

$$\alpha_1 \tilde{s}_1 \cdot \tilde{s}_3 + \alpha_2 \tilde{s}_2 \cdot \tilde{s}_3 + \alpha_3 \tilde{s}_3 \cdot \tilde{s}_3 = +1 \quad (+ve \text{ class})$$

Let's substitute the values for \tilde{s}_1, \tilde{s}_2 and \tilde{s}_3 in the above equations.

$$\alpha_1 \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} + \alpha_2 \begin{pmatrix} 3 \\ 1 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} + \alpha_3 \begin{pmatrix} 3 \\ -1 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} = -1$$

$$\alpha_1(1 + 0 + 1) + \alpha_2(3 + 0 + 1) + \alpha_3(3 + 0 + 1) = -1$$

$$2\alpha_1 + 4\alpha_2 + 4\alpha_3 = -1 \quad \text{----- (i)}$$

$$\alpha_1 \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \begin{pmatrix} 3 \\ 1 \\ 1 \end{pmatrix} + \alpha_2 \begin{pmatrix} 3 \\ 1 \\ 1 \end{pmatrix} \begin{pmatrix} 3 \\ 1 \\ 1 \end{pmatrix} + \alpha_3 \begin{pmatrix} 3 \\ -1 \\ 1 \end{pmatrix} \begin{pmatrix} 3 \\ 1 \\ 1 \end{pmatrix} = 1$$

$$\alpha_1(3 + 0 + 1) + \alpha_2(9 + 1 + 1) + \alpha_3(9 - 1 + 1) = 1$$

$$4\alpha_1 + 11\alpha_2 + 9\alpha_3 = 1 \quad \text{----- (ii)}$$

$$\alpha_1 \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} \begin{pmatrix} 3 \\ -1 \\ 1 \end{pmatrix} + \alpha_2 \begin{pmatrix} 3 \\ 1 \\ 1 \end{pmatrix} \begin{pmatrix} 3 \\ -1 \\ 1 \end{pmatrix} + \alpha_3 \begin{pmatrix} 3 \\ -1 \\ 1 \end{pmatrix} \begin{pmatrix} 3 \\ -1 \\ 1 \end{pmatrix} = +1$$

$$\alpha_1(3 + 0 + 1) + \alpha_2(9 - 1 + 1) + \alpha_3(9 + 1 + 1) = 1$$

$$4\alpha_1 + 9\alpha_2 + 11\alpha_3 = 1 \quad \text{----- (iii)}$$

Simplifying the above 3 equations (i), (ii) and (iii) we get:

$$\alpha_1 = -3.5, \alpha_2 = 0.75 \text{ and } \alpha_3 = 0.75$$

The hyperplane that discriminates the positive class from the negative class is given by

$$\text{Weight Vector } (\tilde{w}) = \sum_i \alpha_i \tilde{S}_i$$

Substituting the values we get:

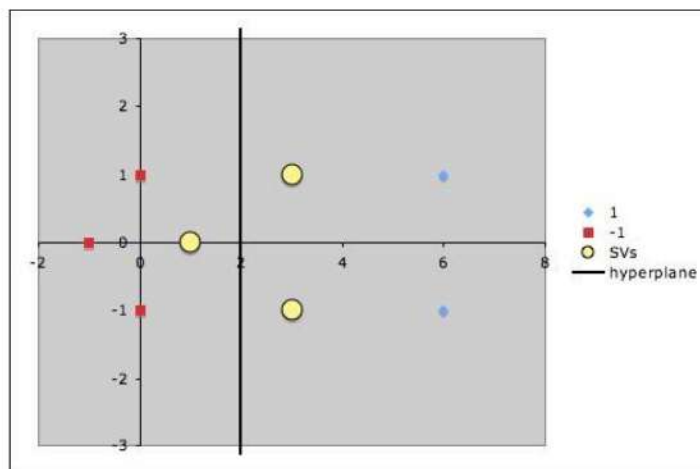
$$\tilde{w} = \alpha_1 \tilde{S}_1 + \alpha_2 \tilde{S}_2 + \alpha_3 \tilde{S}_3$$

$$\tilde{w} = -3.5 \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} + 0.75 \begin{pmatrix} 3 \\ 1 \\ 1 \end{pmatrix} + 0.75 \begin{pmatrix} 3 \\ -1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ -2 \end{pmatrix}$$

Finally, remembering that our vectors are augmented with a bias. Hence we can equate the last entry in \tilde{w} as the hyperplane offset b . Therefore the separating hyperplane equation

$$y = wx + b \text{ with } w = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \text{ and offset } b = -2.$$

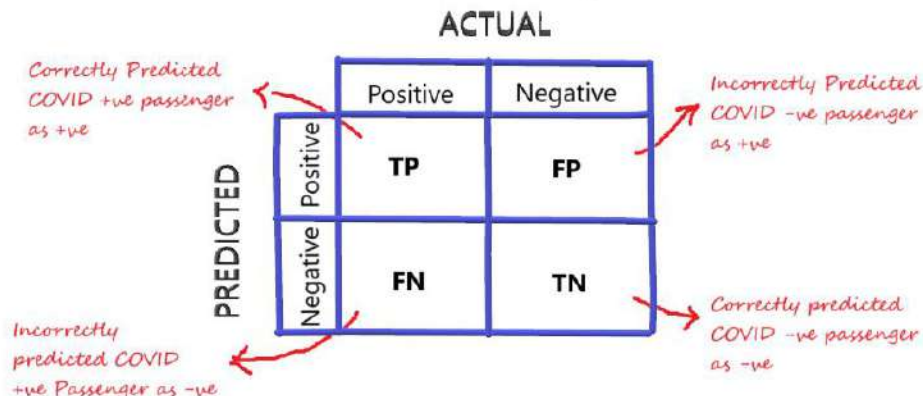
Hence, equation of hyper plane that divides data points is $x - 2 = 0$.



Performance Measures of Classification

Before selecting a classification algorithm to solve a problem, we need to evaluate its performance. Confusion matrix is widely used for computing performance measures of classification algorithm.

A Confusion matrix is an N x N matrix used for evaluating the performance of a classification model, where N is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model. This gives us a holistic view of how well our classification model is performing and what kinds of errors it is making. The structure of confusion matrix N = 2 classes looks like as shown in figure below:



- **True Positive (TP):** It represents correctly classified positive classes. Both actual and predicted class are positive here.
- **False Positive (FP):** It represents incorrectly classified positive classes. These are the positive classes predicted by the model that were actually negative. This is called Type I error.
- **True Negative (TN):** It represents correctly classified Negative classes. Both actual and predicted class are negative here.
- **False Negative (FN):** It represents incorrectly classified negative classes. These are the negative classes predicted by the model that were actually positive. This is called Type II error.

Four widely used performance measures used for evaluating classification models are:

- **Accuracy:** It is the percentage of correct predictions made by the model.
- **Precision:** It is percentage of predicted positives that are actually positive.
- **Recall:** It is the percentage of actual positives that are correctly classified by the model.
- **F1-score:** It is the harmonic mean of recall and precision. It becomes high only when both precision and recall are high.

<p>Accuracy:</p> $ACC = \frac{TP + TN}{TP + TN + FP + FN}$	<p>Recall:</p> $Recall = \frac{TP}{TP + FN}$
<p>Precision:</p> $Precision = \frac{TP}{TP + FP}$	<p>F₁ score:</p> $F_1 = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}}$

Example

Q. “Suppose that we have to classify 100 people (which includes 40 pregnant women and the remaining 60 are not pregnant women and men with a fat belly) as pregnant or not pregnant. Out of 40 pregnant women 30 pregnant women are classified correctly and the remaining 10 pregnant women are classified as not pregnant by the machine learning algorithm. On the other hand, out of 60 people in the not pregnant category, 55 are classified as not pregnant and the remaining 5 are classified as pregnant”. Compute accuracy, precision, recall, and F1-score for the above example.

Solution:

Here,

$$TN = 55, FP = 5, FN = 10, TP = 30.$$

The confusion matrix is as follows:

		ACTUAL	
		Positive	Negative
PREDICTED	Positive	TP = 30	FP = 5
	Negative	FN = 10	TN = 55

Now,

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} = \frac{30+55}{30+55+5+10} = 0.85 = 85\%$$

$$\text{Precision} = \frac{TP}{TP+FP} = \frac{30}{30+5} = 0.857 = 85.7\%$$

$$\text{Recall} = \frac{TP}{TP+FN} = \frac{30}{30+10} = 0.75 = 75\%$$

$$F1 - \text{Score} = \frac{2}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}} = \frac{2 * (\text{Recall} * \text{Precision})}{(\text{Recall} + \text{Precision})} = \frac{2 * (0.75 * 0.857)}{(0.75 + 0.857)} = 0.799$$

Issues Regarding Classification and Prediction

Issue (1): Data Preparation

- **Data Cleaning:** Data cleaning involves removing the noise (by applying smoothing techniques) and treatment of missing values (by replacing a missing value with the most commonly occurring value for that attribute, or with the most probable value based on statistics).
- **Relevance Analysis:** Removing any irrelevant or redundant attributes from the learning process.
- **Data Transformation:** The data can be transformed by any of the following methods:
 - **Normalization:** Normalization involves scaling all values for a given attribute so that they fall within a small specified range.
 - **Generalization:** The data can also be transformed by generalizing it to the higher concept. For this purpose we can use the concept hierarchies.

Issue (2): Comparing Classification Methods

- **Predictive Accuracy:** This refers to the ability of the model to correctly predict the class label of new or previously unseen data.
- **Speed:** This refers to the computation costs involved in generating and using the model or classifier.
- **Robustness:** This is the ability of the model to make correct predictions from given noisy data or data with missing values.
- **Scalability:** This refers to the ability to construct the model efficiently given large amount of data.
- **Interpretability:** This refers to the level of understanding and insight that is provided by the model.

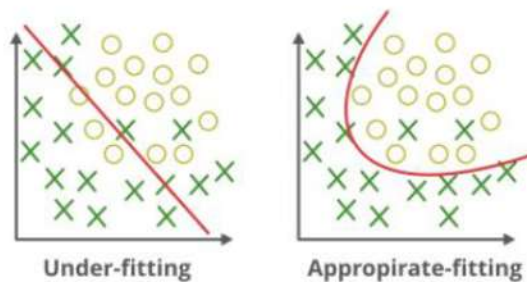
Overfitting and Underfitting

Underfitting

When a model has not learned the patterns in the training data well and is unable to generalize well on the new data, it is known as underfitting. An underfit model has poor performance on the training data and will result in unreliable predictions. Underfitting occurs due to high bias and low variance.

Obviously an underfitted machine learning model is not a suitable model for making predictions because it has poor performance on the training data too.

Example: As shown in figure below, the model is trained to classify between the circles and crosses. However, it is unable to do so properly due to the straight line, which fails to properly classify either of the two classes.



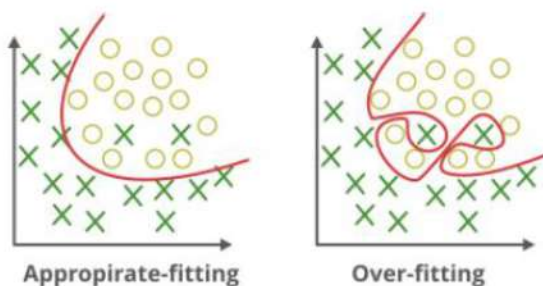
Techniques to reduce underfitting

- Increase model complexity
- Increase the number of features in the dataset
- Reduce noise in the data
- Increase the duration of training the data

Overfitting

When a model performs very well for training data but has poor performance with test data (new data), it is known as overfitting. In this case, the machine learning model learns the details and noise in the training data such that it negatively affects the performance of the model on test data. Overfitting can happen due to low bias and high variance.

Example: As shown in figure below, the model is trained to classify between the circles and crosses, and unlike last time, this time the model learns too well. It even tends to classify the noise in the data by creating an excessively complex model (right).



Techniques to reduce overfitting

- Using K-fold cross-validation
- Reduce model complexity
- Training model with sufficient data
- Remove unnecessary or irrelevant features.
- Using Regularization techniques such as Lasso and Ridge
- Early stopping during the training phase

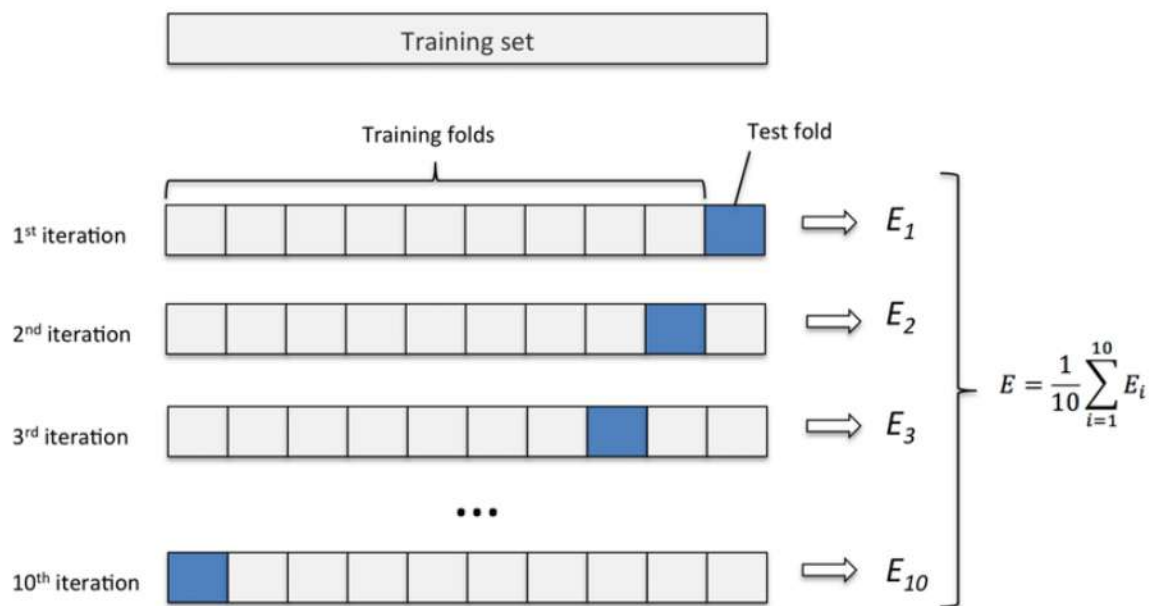
- Overfitting is the result of using an excessively complicated model while Underfitting is the result of using an excessively simple model or using very few training samples.
- A model that is underfit will have high training and high testing error while an overfit model will have extremely low training error but a high testing error.

K-fold Cross Validation

Cross validation is the model validation technique for accessing how the result of the statistical analysis will generalize to an independent data set.

In ***k-fold cross-validation***, the original sample is randomly partitioned into ***k*** equal sized sub samples. Out of the ***k*** subsamples, a single subsample is retained as the validation data for testing the model, and the remaining ***k - 1*** subsamples are used as training data. The cross-validation process is then repeated ***k*** times, with each of the ***k*** subsamples used exactly once as the testing data. The ***k*** results can then be averaged to produce a single estimation.

If ***k = 10***, (10 – fold validation), first divide the data into 10 equal parts, then use 9 parts for training and 1 part for testing.



The advantage of this method is that all observations are used for both training and testing, and each observation is used for testing exactly once.

McNemar's Test

The McNemar Test is a non-parametric statistical test for paired comparison that can be applied to compute the performance of two classifiers. It is based on a 2×2 confusion (contingency) table of the two model's predictions. The layout of 2×2 confusion matrix suitable for McNemar's test is shown in figure below:

		Model 2		
		Correct	Incorrect	
Model 1	Correct	n_{11}	n_{12}	$n_{1\bullet}$
	Incorrect	n_{21}	n_{22}	$n_{2\bullet}$
		$n_{\bullet 1}$	$n_{\bullet 2}$	n_{test}

n_{ii} are the number of concordant pairs, that is, the number of observations that both models classify the same way (correctly or incorrectly). n_{ij} , $i \neq j$, are the number of discordant pairs, that is, the number of observations that models classify differently (correctly or incorrectly).

In McNemar's Test, we formulate the null hypothesis that the probabilities $p(n_{12})$ and $p(n_{21})$ are the same. Thus, the alternative hypothesis is that the performances of the two models are not equal.

The McNemar test statistic ("chi-squared") can be computed as follows:

$$\chi^2 = \frac{(|n_{12} - n_{21}| - 1)^2}{n_{12} + n_{21}}$$

If the sum of cell n_{12} and n_{21} is sufficiently large, the χ^2 value follows a chi-squared distribution with one degree of freedom. After setting a significance threshold, e.g. $\alpha = 0.05$ we can compute the p-value - assuming that the null hypothesis is true, the p-value is the probability of observing this empirical (or a larger) chi-squared value. If the p-value is lower than our chosen significance level, we can reject the null hypothesis that the two model's performances are equal.

Example

		Antibody test		
		Positive	Negative	
RT-PCR test	Positive	55	5	60
	Negative	20	40	60
		75	45	120

Null Hypothesis: There is no difference in outcome of 2 test method: $P(A) = P(B)$

Alternative Hypothesis: There is difference in outcome of 2 test method: $P(A) \neq P(B)$

Where, A & B are discordant pairs.

Significance level: $\alpha = 0.05$

Test statistics:

$$\chi^2 = \frac{(|5-20|-1)^2}{5+20} = 7.84$$

Calculating P-value:

$$0.005 < p - \text{value} < 0.01$$

Here, the p-value is lower than our chosen significance level (0.05) so we reject null hypothesis.

Prediction

Regression analysis can be used to model the relationship between one or more independent or **predictor** variables and a dependent or **response** variable. In the context of data mining, the predictor variables are the attributes of interest describing the tuple. In general, the values of the predictor variables are known. The response variable is what we want to predict. Given a tuple described by predictor variables, we want to predict the associated value of the response variable.

There are three categories of regression:

1. **Linear Regression:** It is used to estimate relationship between variables by fitting linear equation to observe the data. Linear regression analysis involves a response variable y and a single predictor variable x . The simplest form of regression is

$$y = a + bx$$

Where y is response variable and x is single predictor variable. y is a linear function of x . a and b are regression coefficients.

Let D be a training set consisting of values of predictor variables, x , for some population and their associated values for response variable, y . The training set contains $|D|$ data points of the form $(x_1, y_1), (x_2, y_2), \dots, (x_{|D|}, y_{|D|})$.

The regression coefficients can be estimated using this method with the following equations:

$$b = \frac{\sum_{i=1}^{|D|} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{|D|} (x_i - \bar{x})^2}, \quad a = \bar{y} - b\bar{x}$$

2. **Multiple Linear Regression:** Multiple linear regression is an extension of linear regression so as to involve more than one predictor variable. It allows response variable y to be modelled as a linear function of, say, n predictor variables or attributes A_1, A_2, \dots, A_n , describing a tuple, X . (that is, $X = x_1, x_2, \dots, x_n$).

Our training data set, D , contains data of the form $(x_1, y_1), (x_2, y_2), \dots, (x_{|D|}, y_{|D|})$, where the X_i are the n -dimensional training tuples with associated class labels, y_i . An example of a multiple linear regression model based on two predictor attributes or variables, A_1 and A_2 , is

$$y = a_0 + a_1x_1 + a_2x_2$$


Where x_1 & x_2 are the values of attributes A_1 and A_2 , respectively, in X and a_0, a_1, a_2 are regression coefficient.

3. **Non Linear Regression:** It is also known as polynomial regression. Polynomial regression is often of interest when there is just one predictor variable. It can be modelled by adding polynomial terms to the basic linear model. By applying transformations to the variables, we can convert the nonlinear model into a linear one that can be solved by the method of least squares.

Consider a cubic polynomial relationship given by $y = a_0 + a_1x + a_2x^2 + a_3x^3$. To convert this equation to linear form, we define new variables: $x_1 = x$, $x_2 = x^2$, $x_3 = x^3$

$$y = a_0 + a_1x_1 + a_2x_2 + a_3x_3$$

Which is easily solved by the method of least squares using software for regression analysis. Polynomial regression is a special case of multiple regression.



Please let me know if I missed anything or
anything is incorrect.
poudeljayanta99@gmail.com