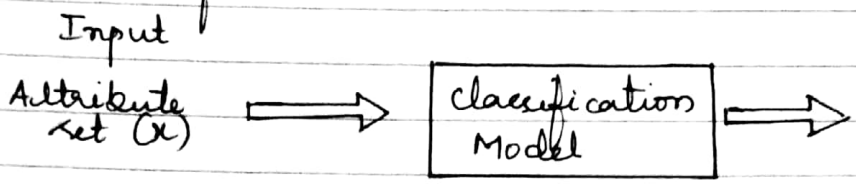
UNIT - 3 : CLASSIFICATION

Date No

Classification is called Supervised Learning

Definition :

Classification is the task of learning a target function f that maps each attribute set x to one of the predefined class labels y. The target function is informally known as a classification model



Given a collection of records (training set), Each record contains a set of attributes, one of the attributes is the class. find a model for class attribute as a function of the values of other attributes

Goal :- Previously unseen records should be assigned a class as accurately as possible

A test set is used to determine the accuracy of the model. Usually, the given data set is divided into training and test sets, with training set used to build the model and test set used to validate it.

Examples of classification Task

- Predicting the tumor cells are benign or malignant
- classifying creditcard transactions as legitimate or fraudulent
- Categorizing news stories as entertainement, sports, weather etc

Date No

General approach to solving a classification Problem

A classification technique (or classifier) is a systematic approach to building classification models from an input data set

Eg: decision tree classifiers, Rule-Based classifiers, Neural Networks, Support Vector Machines, and Naive Bayes classifiers

Each technique employs a learning algorithm to identify a model that best fits the relationship between the attribute set and class label of the input data

The model generated by a learning algorithm should both fit the input data well and correctly predict the class labels of the records it has never seen before Therefore, a key objective of the learning algorithm is to build models with good generalization capability i.e., models that accurately predict the class labels of previously unknown records

Preliminaries

The input data for a classification task is a collection of records. Each record also known as an instance or example.

classification is the task of learning a target function f that maps each attribute set x to one of the predefined class labels y

The target function is also known informally as a classification model.

Date No

A classification model is useful for the following purposes.
- Descriptive Modeling
- Predictive Modeling

Descriptive Modeling

A classification model can serve as an explanatory tool to distinguish between objects of different classes.

For eg:- it would be useful - for both biologists and others - to have a descriptive model that summarizes the data and explains what features define a vertebrate as a mammal, reptile, bird, fish or amphibian.

Predictive Modeling

A classification model can also be used to predict the class label of unknown records. A classification model can be treated as a black box that automatically assigns a class label when presented with the attribute set of an unknown record.

* classification techniques are most suited for predicting or describing data sets with binary or nominal categories.

Confusion Matrix

A Confusion matrix contains information about actual and predicted classifications done by a classification system. Performance of such systems is commonly evaluated using the data in the matrix.

		Predicted		
$n = 165$		NO	YES	
Actual	NO	50	10	60
	YES	5	100	105
		55	110	

Date No.

- There are two possible predicted classes: "Yes" and "No". If we were predicting the presence of a disease, "Yes" would mean they have the disease. and "No" would mean they don't have the disease
- The classifier made a total of 165 predictions (Eg:- 165 patients were being tested for the presence of that disease)
- out of those 165 cases, the classifier predicted "Yes" 110 times and 'No' 55 times
- (we predicted N)
- In reality, 105 patients in the sample have the disease, and 60 patients do not.

Basic terms of Confusion matrix

- true positives (TP):
  Predicted Yes, Actual Yes
- true negatives (TN):
  Predicted No, Actual No
- false positives (FP):
  Predicted Yes, Actual No
- false negatives (FN):
  Predicted No, Actual Yes

Although a Confusion matrix provides the information needed to determine how well a classification model performs, summarizing this information with a single number would make it more convenient to compare the performance of different models. This can also be

done using a performance metric such as
- Accuracy
- Error rate

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} = \frac{(TP + TN)}{\text{Total no. of predictions}}$$

$$\text{Error rate} = \frac{\text{Number of wrong predictions}}{\text{Total number of predictions}} = \frac{(FP + FN)}{\text{Total no. of predictions}}$$

Decision Tree Induction

Decision tree algorithm belongs to the family of supervised learning algorithms. Unlike other supervised learning algorithms, decision tree algorithm can be used for solving regression and classification problems too.

The general motive of using Decision tree is to create a training model which can use to predict class or value of target variables by learning decision rules inferred from prior data (training data)

The decision tree algorithm tries to solve the problem, by using tree representation. Each internal node of the tree corresponds to an attribute, each leaf node corresponds to a class label.

Decision Tree can handle both categorical and numerical data

The topmost decision node in a tree which corresponds to the best predictor called rootnode.

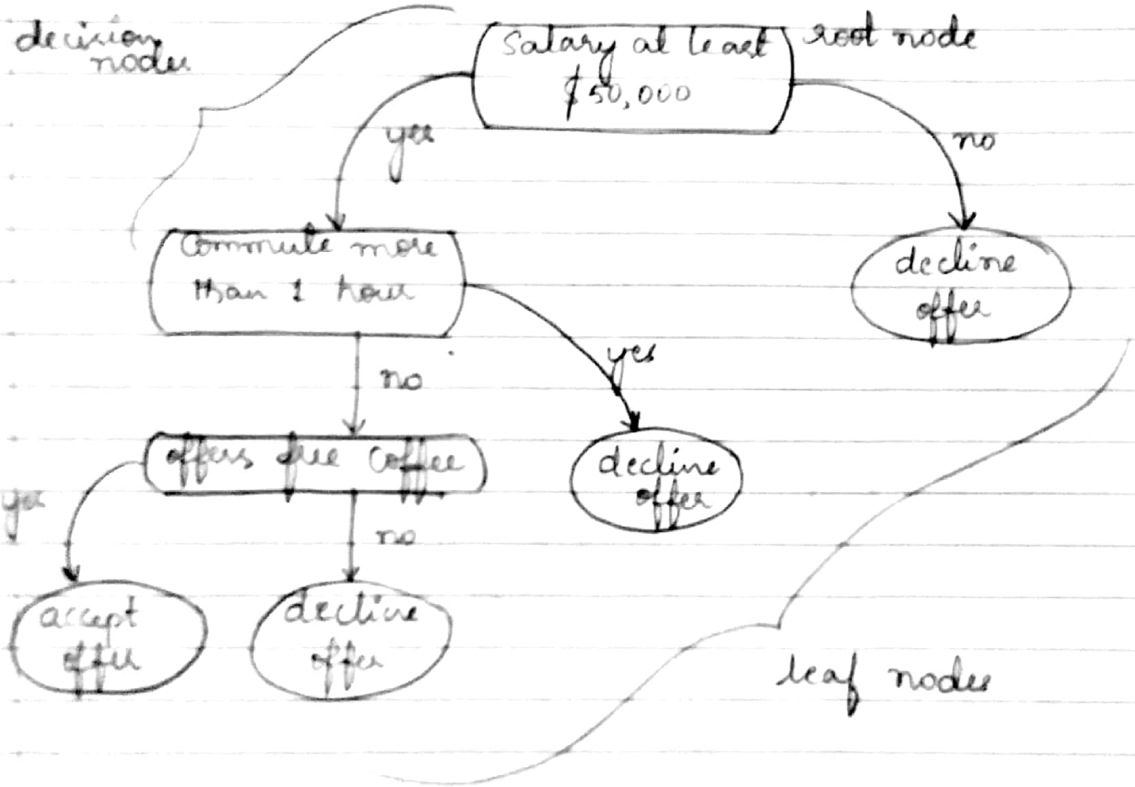
MATRIKAS

How to Build a Decision Tree

Decision tree algorithm pseudocode:

1. Place the best attribute of the dataset at the root of the tree.
2. Split the training set into subsets.
3. Repeat step 1 and step 2 on each subset until you find leaf nodes in all the branches of the tree.

Decision Tree: Should I accept a new job offer?



Decision Tree classifier

Efficient algorithms have been developed to induce a reasonably accurate, albeit suboptimal decision tree in a ~~reasonable~~ (even if) reasonable amount of time. These algorithms usually employ a greedy strategy that grows a decision tree by making a series of locally optimum decisions about which attribute to use for partitioning the data.

Date No

One such algorithm is Hunt's Algorithm, which is the basis of many existing decision tree induction algorithms including ID3, C4.5 and CART.

Hunt's Algorithm

It employs a top-down search, greedy search through the space of possible decision trees.

General structure of Hunt's Algorithm

- Let D_t be the set of training records that reach a node t.

- **General Procedure:**
  - if D_t contains records that belong the same class y_t, then t is a leaf node labeled as y_t.
  - if D_t contains records that belong to more than one class, use an attribute test to split the data into smaller subsets. Recursively apply the procedure to each subset.

	binary	categorical	continuous	class
TID	Home owner	Marital status	Annual income	Defaulted Borrower
1	Yes	Single	125K	NO
2	NO	Married	100K	NO
3	NO	Single	70K	NO
4	Yes	Married	120K	NO
5	NO	Divorced	95K	Yes
6	NO	Married	60K	NO
7	Yes	Divorced	220K	NO
8	NO	Single	85K	Yes
9	NO	Married	75K	NO
10	NO	Single	90K	Yes

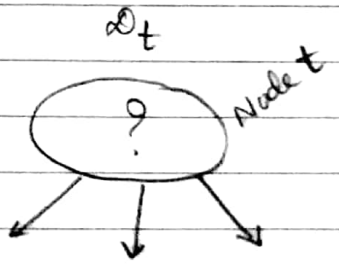


fig:- Training set for predicting borrowers who will default on loan payment.

- which attribute should be tested at each splitting node?
  - Use some heuristic.

Date No

Tree Induction

Issues:

- Determine when to stop splitting
- Determine how to split the records
  * which attribute to use in a split node split?
    → How to determine the best split?
  * How to specify the attribute test condition?
    Eg:- $x < 1$? or $x + y < 1$?
  * 2-way split? or multi-way split?

Methods for Expressing Attribute Test Conditions

Decision tree induction algorithms must provide a method for expressing an attribute test condition and its corresponding outcomes for different attribute types.

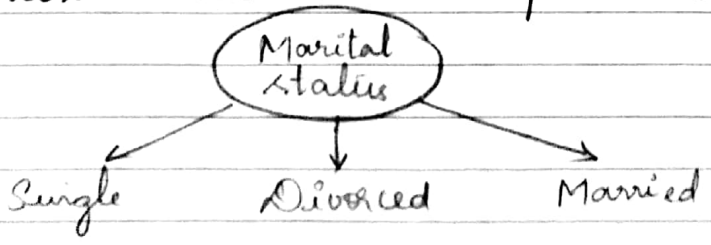
1. Binary Attributes

The test condition for a binary attribute generates two potential outcomes

fig: Test condition for binary attributes

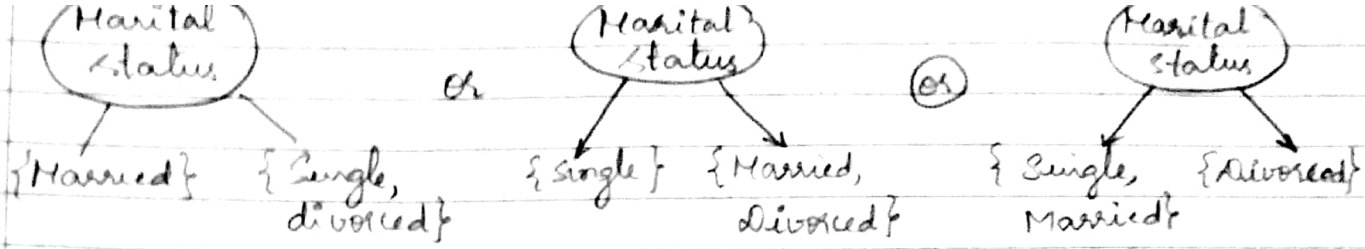
2. Nominal Attributes

Since nominal attribute can have many values, its test condition can be expressed in two ways.



(a) Multi-way split

MATRIKAS



(b) Binary split {by grouping attribute values}

fig: Test conditions for nominal attributes

3. Ordinal attributes

Ordinal attributes can also produce binary or multiway splits.

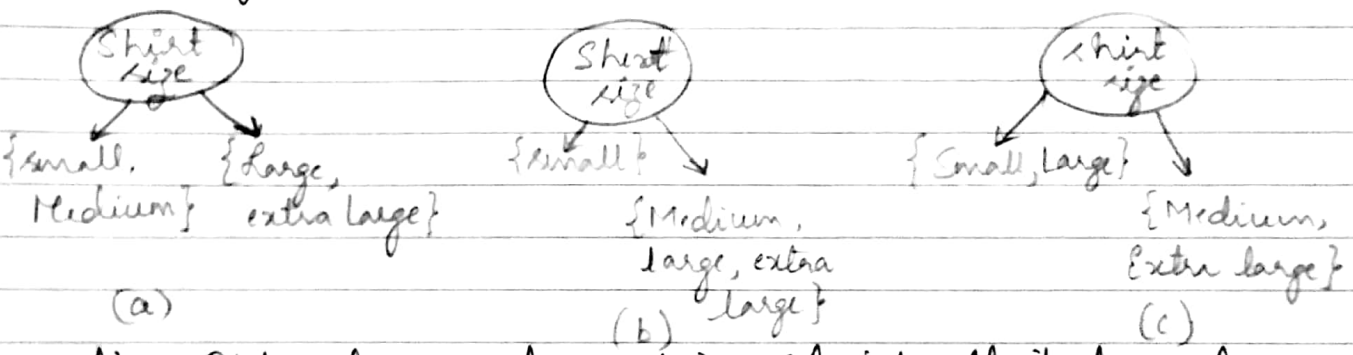


fig: Different ways of grouping ordinal attribute values.

4. Continuous Attributes

For Continuous attributes, the test condition can be expressed as a comparison test $(A < v)$ or $(A \geq v)$ with binary outcomes or a range query with outcomes of the form $v_i \leq A < v_{i+1}$ for $i = 1, 2, \ldots, k$.

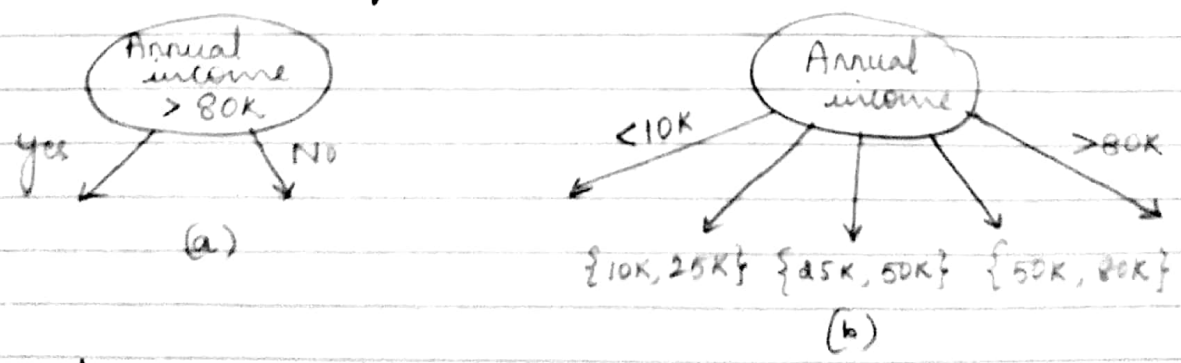


fig: - Test Condition for Continuous attributes

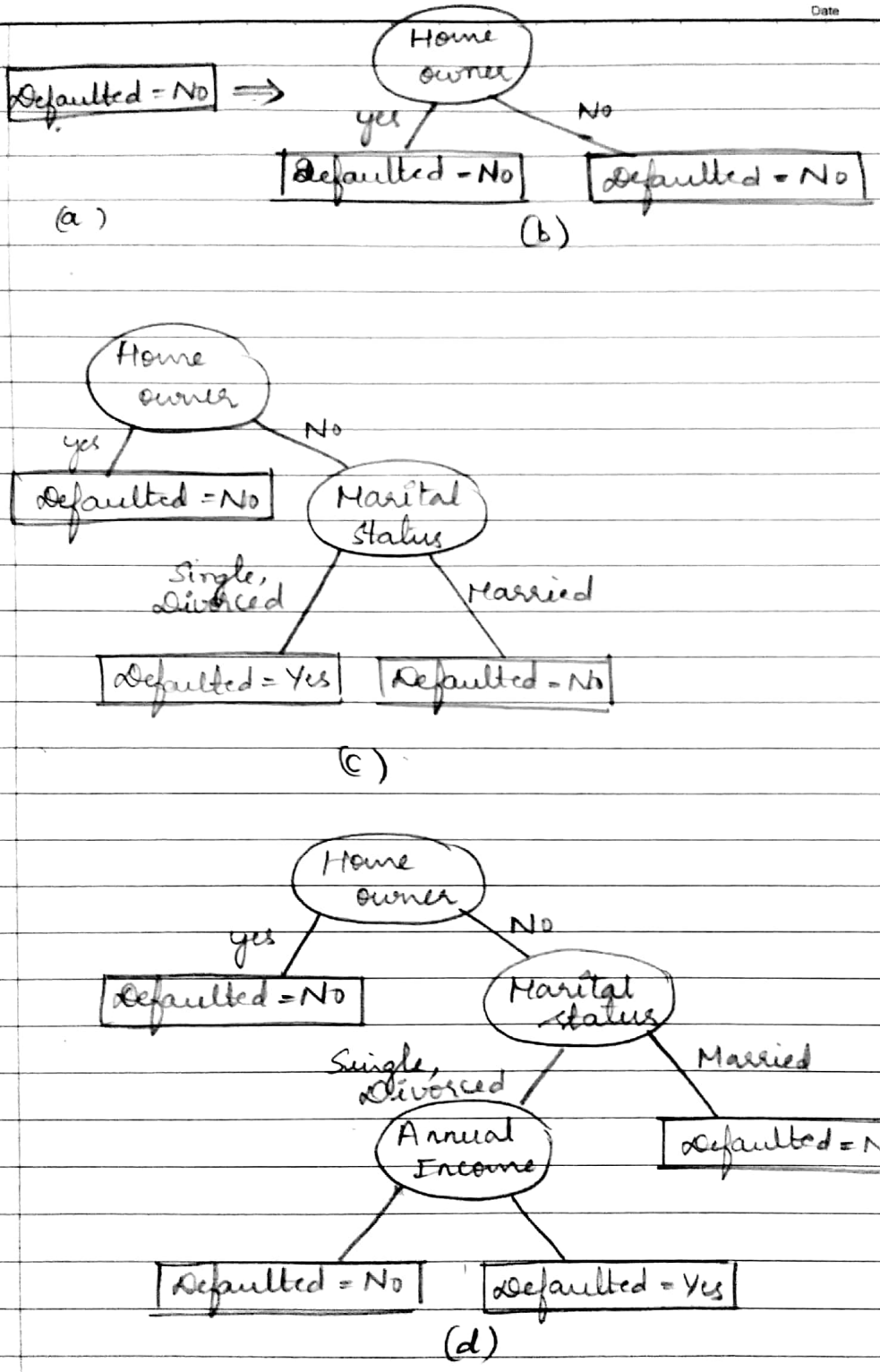


fig: Hunt's Algorithm for inducing decision trees

Impurity measures include
- Entropy
- Gini
- Classification error

Terms used

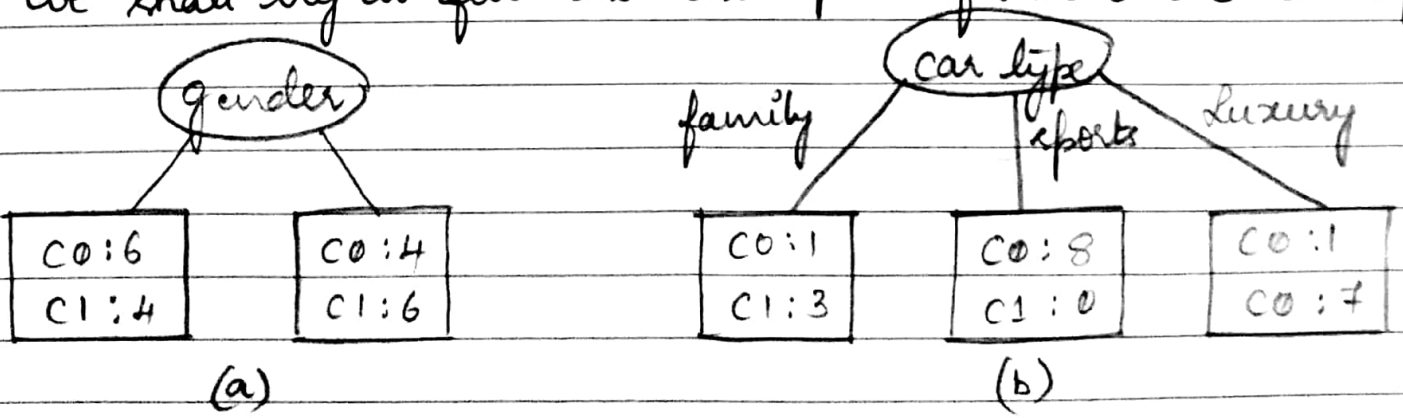
c ⇒ no. of classes

t ⇒ total no. of records to be considered for a node.

P(i/t) ⇒ fraction of records belonging to class i at a given node t.

Eg:- Consider 20 records of which 10 records belong to class 0 (C0) & 10 records belonging to class 1 (C1)

we shall try to find the best split for the above example



(b) is considered as pure set or homogeneous as in sports category we have pure set & also max^m results are on one side
So, split on car type is best

The measures developed for selecting the best split are often based on the degree of impurity of the child nodes.
The impurity measures include

$$\text{Entropy}(t) = \sum_{i=0}^{C-1} p(i|t) \log_2 p(i|t)$$

where,
$t \rightarrow$ node , $C \rightarrow$ no. of classes

$$\text{Gini}(t) = 1 - \sum_{i=0}^{C-1} [p(i|t)]^2$$

Date No

All the three measures attain their maximum value when class distribution is uniform. i.e., when P=0.5. Max^m values for the measures are attained when all the records belong to the same class (i.e., when P=0 or 1)

Compute the different Impurity Measures

①

Node N_1	Count
Class = 0	0
Class = 1	6

$$\text{Gini} = 1 - \left(\left(\frac{0}{6}\right)^2 + \left(\frac{6}{6}\right)^2\right) = 1 - (0+1) = 0$$

$$\text{Entropy} = -\frac{0}{6}\log_2\left(\frac{0}{6}\right) - \frac{6}{6}\log_2\left(\frac{6}{6}\right) \quad | \quad \log_2 1 = 0$$

$$= 0 - 1\log_2 1 = 0$$

$$\text{Error} = 1 - \max\left[\frac{0}{6}, \frac{6}{6}\right]$$
$$= 1 - \max(0, 1)$$
$$= 1 - 1 = 0$$

②

Node N_2	Count
class = 0	1
Class = 1	5

$$\text{Gini} = \left(\frac{1}{6}\right)^2 - \left(\frac{5}{6}\right)^2$$
$$= 1 - \frac{1}{36} - \frac{25}{36} = \frac{10}{36} = 0.278$$

$$\text{Entropy} = \frac{1}{6}\log_2\left(\frac{1}{6}\right) - \frac{5}{6}\log_2\left(\frac{5}{6}\right) = 0.650$$

$$\text{Error} = 1 - \max\left[\frac{1}{6}, \frac{5}{6}\right] = 0.167$$

③

Node N_3	Count
Class = 0	3
Class = 1	3

$$\text{Gini} = 1 - \left(\frac{3}{6}\right)^2 - \left(\frac{3}{6}\right)^2 = 0.5$$

$$\text{Entropy} = \frac{3}{6}\log_2\left(\frac{3}{6}\right) - \frac{3}{6}\log_2\left(\frac{3}{6}\right) = 1$$

MATRIKAS

Error = $1 - \max\left(\frac{3}{6}, \frac{3}{6}\right) = 1 - \frac{1}{2} = 0.5$

* Node N_1 has the lowest impurity & N_3 has the highest

Algorithm for Decision Tree Induction / TreeGrowth

```
TreeGrowth(E, F)
1: If stopping_cond(E, F) = true, then
2:     leaf = createNode().
3:     leaf.label = classify(E).
4:     return leaf.
5: else
6:     root = createNode()
7:     root.test_cond = find_best_split(E, F)
8:     let V = {v | v is a possible outcome of root.test_cond}.
9:     for each v ∈ V do
10:        Ev = {e | root.test_cond(e) = v and e ∈ E}.
11:        child = TreeGrowth(Ev, F)
12:        add child as descendent of root and label the
           edge(root → child) as v.
13:    end for
14: end if
15: return root.
```

Explanation

The input to this algorithm consisting of
E - The training records
F - The attribute set

1) The createNode() function extends the decision tree by creating a new node. A node in the decision tree has either a test condition, denoted as node.test_cond, or a class label denoted as node.label

2) The find_best_split() function determines which

attribute should be selected as the test condition for splitting the training records

The choice of the test condition depends on which impurity measure is used to determine the goodness of a split.

3) The classify() function determines the class label to be assigned to a leaf node

4) The stopping_cond() function is used to terminate the tree-growing process by testing whether all the records have either the same class label or the same attribute values.

Characteristics of Decision Tree Induction

Decision tree induction is a nonparametric approach for building classification models. In other words, it does not require any prior assumptions regarding the type of probability distributions satisfied by the class and other attributes.

Finding an optimal decision tree is an NP-Complete problem. Many decision tree algorithms employ a heuristic-based approach to guide their search in the vast hypothesis space

Techniques developed for constructing decision trees are computationally inexpensive, making it possible to quickly construct models even when the training set size is very large.

Decision trees, especially an expressive representation for learning dis-

4. Decision trees, especially smaller-sized trees, are relatively easy to interpret

5. Decision trees provide an expressive representation for learning discrete valued functions.

6. Decision tree algorithms are quite robust to the presence of noise, especially when methods for avoiding overfitting

7. The presence of redundant attributes doesnot adversely affect the accuracy of decision trees.

8. Since most of the decision tree algorithms employ a top-down, recursive partitioning approach, the number of records becomes smaller as we traverse down the tree.

9. A subtree can be replicated multiple times in a decision tree as given below

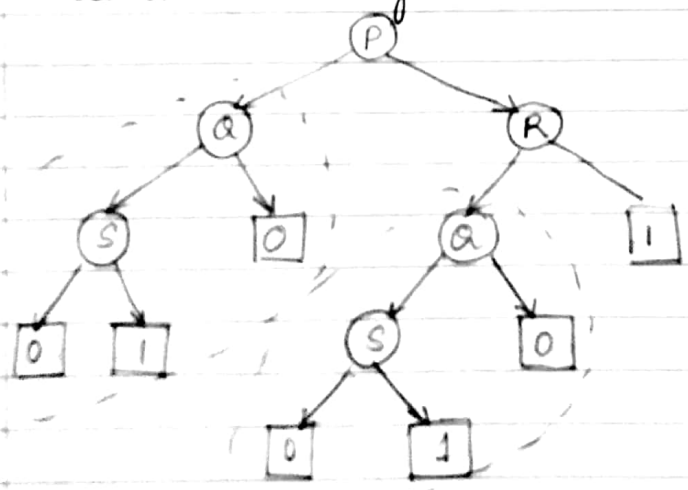


fig: Tree replication problem. The same subtree can appear at different branches

This makes the decision tree more complex than necessary and perhaps more difficult to interpret.

the test conditions involve using only a single attribute at a time.

Evaluating the Performance of a classifier

It is often useful to measure the performance of the model on the test set because such a measure provides an unbiased estimate of its generalization error.

The accuracy or error rate computed from the test set can also be used to compare the relative performance of different classifiers on the same domain. In order to do this, the class labels of the test records must be known.

Holdout Method

In this method, the original data with labeled examples is partitioned into two disjoint sets, called the training and the test (test) sets, respectively. A classification model is then induced from the training set and its performance is evaluated on the test set.

Random Subsampling

The holdout method can be repeated several times to improve the estimation of a classifier's performance

Cross-Validation

An alternative to random subsampling is cross-validation. In this approach, each record is used the same number of times for training and exactly once for testing

Bootstrap

In this approach, the training records are sampled with replacement i.e., a record already chosen for training is put back into the original pool of records so that it is equally likely to be redrawn.

Date No.

Classification: Alternative Techniques

Rule-Based Classifier

A rule-based classifier is a technique for classifying records using a collection of "if.... then...." rules.

IF condition THEN conclusion

Let us consider a rule R1

R1: If age = youth AND student = yes
THEN buy-computer = yes

NOTE:

* The IF part of the rule is called rule antecedent or precondition.
* The THEN part of the rule is called rule consequent
* The antecedent part the condition consist of one or more attribute tests and these tests are logically ANDed.
* The consequent part consists of class prediction.

We can also write rule R1 as follows

R1: ((age = youth) ∧ (student = yes)) → (buys-computer = yes)

Rule Extraction

Here we will learn how to build a rule-based classifier by extracting IF-THEN rules from a decision tree.

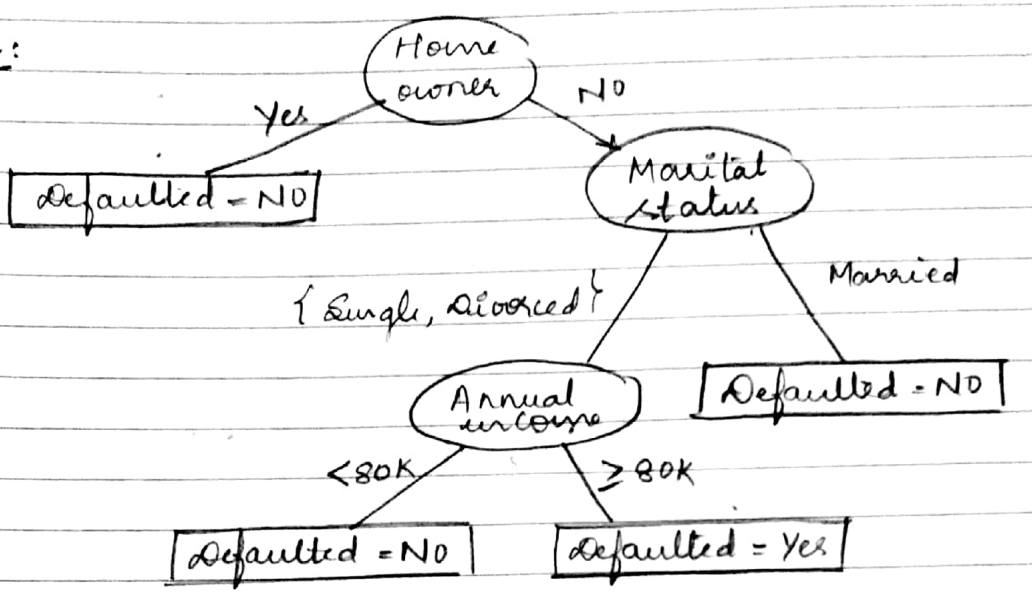
NOTE:

To extract rule from decision tree

* One rule is created for each path from the root to leaf node
* To form a rule antecedent, each splitting criterion is logically ANDed.

* The leaf node holds the class prediction, forming the rule consequent.

Eg:



Rules for the above example can be written as

R1: IF Homeowner = Yes THEN defaulted = NO

R2: IF Homeowner = NO AND Marital status = {Single, divorced} AND Annual income < 80K THEN defaulted = NO

R3: IF Homeowner = NO AND Marital status = Married THEN defaulted = NO

R4: IF Home owner = NO AND Marital status = {Single, Divorced} AND Annual income ≥ 80K THEN defaulted = Yes.

Nearest - Neighbor classifiers

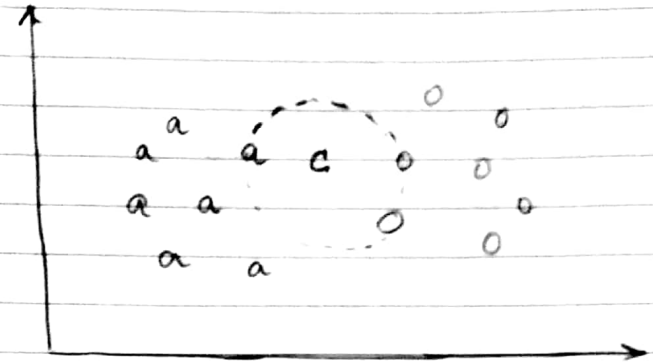
K- Nearest -Neighbor (KNN) Algorithm

In KNN classification, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its K nearest neighbors

The KNN algorithms is to predict the target label by finding the nearest neighbor class. The closest class will be identified using the distance measures like Euclidean distance.

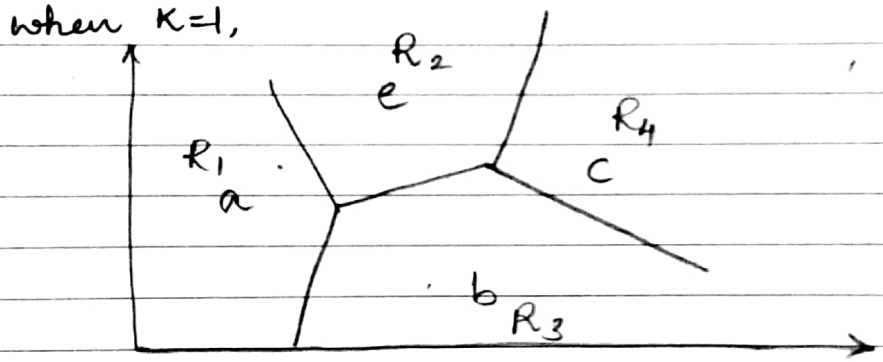
MATRIKAS

Given N training vectors, KNN algorithm identifies the K nearest neighbors of 'c' regardless of labels



Example
- K=3
- Classes 'a' and 'o'
- find class for 'c'

we have 2 votes for 'o', 1 vote for 'a'
In this case, the class of the element 'c' is going to be 'o'



$$R_i = \{ x : d(x, x_i) < d(x, x_j),\ i \neq j \}$$

Advantages of KNN algorithm
- Simple to implement
- Executes quickly for small training data set
- Doesn't need any prior knowledge about the structure of data in the training set
- No retraining is required if the new training pattern is added to the existing training set

Limitation of KNN algorithm
- when the training set is large, it may take a

lot of space
- for every test data, the distance should be computed between test data and all the training data. Thus a lot of time may be needed for the testing.

Naive Bayes classifier

The Naive Bayes Algorithm is a Machine Learning algorithm for classification problems. It is primarily used for text classification, which involves high-dimensional training data sets.

Eg:- Spam filtering, sentimental analysis, classifying new articles.

Main features

* It is simple
* Effective
* We can build models fast
* Make quick predictions

It is a probabilistic classifier. The Naive Bayes algorithm is called "Naive" because it makes the assumption that the occurence of a certain feature is independent of the occurence of other features

Bayes theorem

Bayes' theorem is stated as Probability of the event B given A is equal to the probability of the event A given B multiplied by the probability of A upon probability of B (Conditional Probability)

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

(likelihood)

$P(A|B)$: Probability of occurence of event A given the event B is true.

MATRIKAS

$P(A)$ and $P(B)$: Probabilities of the occurence of event A and B respectively.

$P(B|A)$: Probability of the occurence of event B given the event A is true

The main aim in the Naive Bayes Algorithm is to calculate the conditional probability of an object with a feature vector $x_1, x_2, \cdots, x_n$ belongs to a particular class C_i.

$$P(C_i | x_1, x_2, \cdots, x_n) = \frac{P(x_1, x_2, \cdots, x_n | C_i) \cdot P(C_i)}{P(x_1, x_2, \cdots, x_n)} \quad \text{for } 1 \le i \le k$$

The conditional probability term, $P(x_j | x_{\{j+1\}}, \cdots, x_n, C_i)$ becomes $P(x_j | C_i)$ because of the assumption that features are indipendent

Person ID	Name	Gender	Height	class
1	A	Female	1.6	Short
2	B	Male	2.0	Tall
3	C	Female	1.9	Medium
4	D	Female	1.85	Medium
5	E	Male	2.8	Tall
6	F	Male	1.7	Short
7	G	Male	1.8	Medium
8	H	Female	1.6	Short
9	I	Female	1.65	Short

Question

Using Bayesian classification, classify the following tuple $t = \{$ Alexander, M, 2.2m $\}$

Range of Height	
[0 - 1.6]	[1.81 - 1.9]
[1.61 - 1.7]	[1.91 - 2.0]
[1.71 - 1.8]	[2.0 - ∞]

Gender
Male
Female

Attribute probability

P(short) = 4/9

P(Medium) = 3/9

P(Tall) = 2/9

Attribute		value			Probability		
Gender		short	Medium	Tall	Short	Medium	Tall
	Male	1	1	2	1/4	1/3	2/2
	female	3	2	0	3/4	2/3	0/2
Height							
	[0 - 1.6]	2	0	0	2/4	0	0
	[1.61 - 1.7]	2	0	0	2/4	0	0
	[1.71 - 1.8]	0	1	0	0	1/3	0
	[1.81 - 1.9]	0	2	0	0	2/3	0
	[1.91 - 2.0]	0	0	1	0	0	1/2
	[2.01 - ∞]	0	0	1	0	0	1/2

Step 1:

Findout the probability of the tuple with the class attribute

$P(t|short) = P[M|short] * P([2.0 - \infty]|short)$

$= 1/4 * 0 = \underline{0}$

$P(t|Medium) = P[M|Medium] * P([2.0 - \infty]|Medium)$

$= 1/3 * 0 = \underline{0}$

$P(t|Tall) = P[M|Tall] * P([2.0 - \infty]|Tall]$

$= 1 * 1/2 = \underline{0.5}$ or $\underline{1/2}$

Step 2:

Findout likelihood

Likelihood for short = $P(t|short) * P(short)$

$= 0 * 4/9$

$= \underline{0}$

Likelihood for medium = $P(t|medium) * P(medium)$
$= 0 * 3/9 = 0$

Likelihood for Tall = $P(t|Tall) * P(Tall)$
$= 1/2 * 2/9 = 1/9 = 0.11$

Estimate value → Sum of all the likelihood
$0 + 0 + 0.11$
$P(t) = 0.11$

Step 3:
Finding out the actual probability

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

$$P(Short|t) = \frac{P(t|short) * P(short)}{P(t)}$$
$$= \frac{0 * 4/9}{0.11} = 0$$

$$P(Medium|t) = \frac{P(t|Medium) * P(Medium)}{P(t)}$$
$$= \frac{0 * 3/9}{0.11} = 0$$

$$P(Tall|t) = \frac{P(t|Tall) * P(Tall)}{P(t)}$$
$$= \frac{1/2 * 2/9}{0.11} = \frac{0.11}{0.11} = 1$$

Compare all the probabilities - tall has the highest value. Thus, the class of the given tuple is Tall