MODULE III

CHAPTER 3

Classification

University Prescribed Syllabus w.e.f Academic Year 2021-2022

Basic Concepts; Classification methods : 1. Decision Tree Induction: Attribute Selection Measures, Tree pruning. 2. Bayesian Classification: Naïve Bayes Classifier. Prediction : Structure of regression models; Simple linear regression, Accuracy and Error measures, Precision, Recall, Holdout, Random Sampling, Cross Validation, Bootstrap, Introduction of Ensemble methods, Bagging, Boosting, AdaBoost and Random forest.

Self-learning Topics : Multiple linear regression, logistic regression, Random forest, nearest neighbour classifier, SVM.

3.1 INTRODUCTION

- There are two forms of data analysis that can be used for extracting models describing important classes or to predict future data trends.
- These two forms are as follows :

(i) Classification (ii) Prediction

- Classification is a type of data analysis in which models defining relevant data classes are extracted.
- Classification models, called classifier, predict categorical class labels; and prediction models predict continuous valued functions.
- 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.

3.2 BASIC CONCEPTS

In this section we will discuss what classification is, working of classification, issues in classification and criteria for comparing the methods of classification.

3.2.1 What is Classification ?

- Following are the examples of cases where the data analysis task is Classification –

(i) A bank loan officer wants to analyze the data in order to know which customer (loan applicant) are risky or which are safe.

(ii) A marketing manager at a company needs to analyze a customer with a given profile, who will buy a new laptop.

- In both of the above examples, a model or classifier is constructed to predict the categorical labels. These labels are "risky" or "safe" for loan application data and "yes" or "no" for marketing data.

3.2.2 How Does Classification Work ?

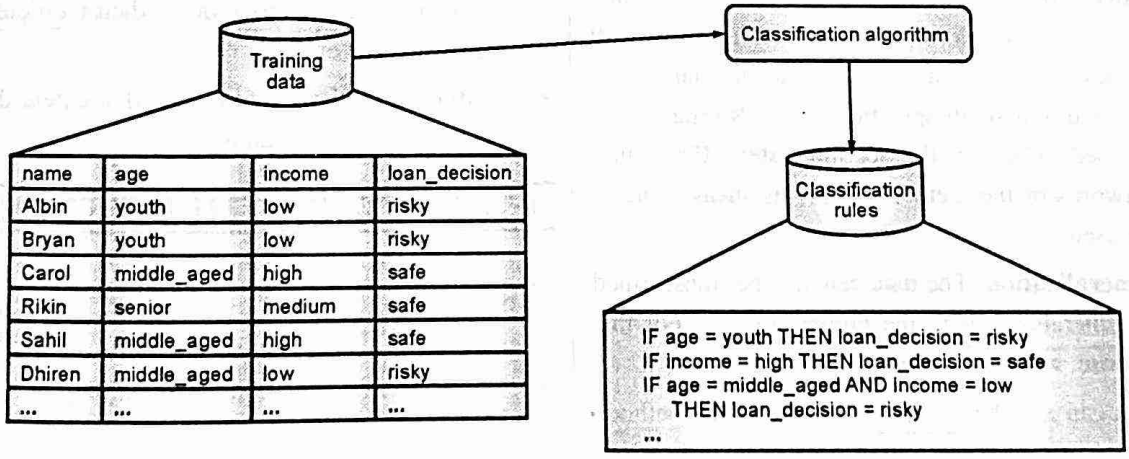
With the help of the bank loan application that we have discussed above, let us understand the working of classification. The Data Classification process includes two steps –

1. Building the Classifier or Model
2. Using Classifier for Classification

1. Building the Classifier or Model

- This step is the learning step or the learning phase.
- In this step the classification algorithms build the classifier.
- The classifier is built from the training set made up of database tuples and their associated class labels.
- Each tuple that constitutes the training set is referred to as a category or class. These tuples can also be referred to as sample, object or data points.

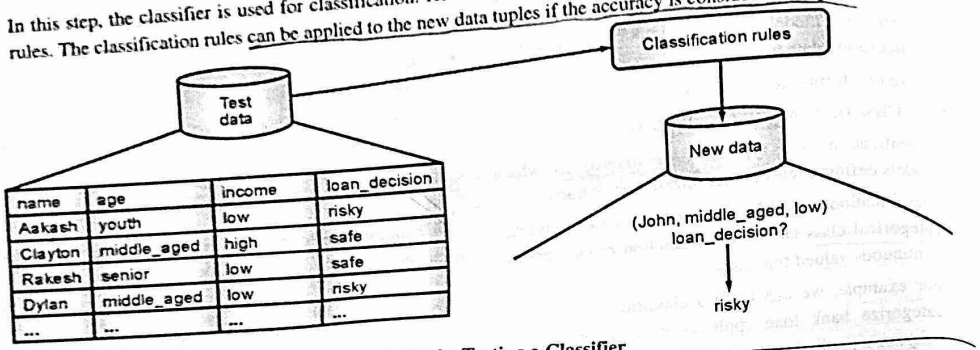
Module 3



(1C1)**Fig. 3.2.1 : Building a Classifier**

2. Using Classifier for Classification

In this step, the classifier is used for classification. Here the test data is used to estimate the accuracy of classification rules. The classification rules can be applied to the new data tuples if the accuracy is considered acceptable.



name	age	income	loan_decision
Aakash	youth	low	risky
Clayton	middle_aged	high	safe
Rakesh	senior	low	safe
Dylan	middle_aged	low	risky
...	...	...	...

(1C2)Fig. 3.2.2 : Testing a Classifier

3.2.3 Classification Issues

The major issue is preparing the data for Classification. Preparing the data involves the following activities:

- **Data Cleaning :** Data cleaning involves removing the noise and treatment of missing values. The noise is removed by applying smoothing techniques and the problem of missing values is solved by replacing a missing value with most commonly occurring value for that attribute.
- **Relevance Analysis :** Database may also have the irrelevant attributes. Correlation analysis is used to know whether any two given attributes are related.
- **Data Transformation and Reduction :** The data can be transformed by any of the following methods.
  - (i) **Normalization :** The data is transformed using normalization. Normalization involves scaling all values for given attribute in order to make them fall within a small specified range. Normalization is used when in the learning step, the neural networks or the methods involving measurements are used.
  - (ii) **Generalization:** The data can also be transformed by generalizing it to the higher concept. For this purpose, we can use the concept hierarchies.
  - (iii) Data can also be reduced by some other methods such as wavelet transformation, binning, histogram analysis and clustering.

3.2.4 Comparison of Classification Methods

Here are the criteria for comparing the methods of Classification.

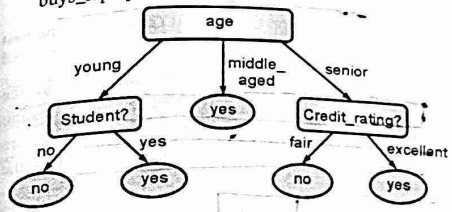
- **Accuracy :** Accuracy of classifier refers to the ability of classifier. It predicts the class label correctly and the accuracy of the predictor refers to how well a given predictor can guess the value of predicted attribute for a new data.
- **Speed :** This refers to the computational cost in generating and using the classifier or predictor.
- **Robustness :** It refers to the ability of classifier or predictor to make correct predictions from given noisy data.
- **Scalability :** Scalability refers to the ability to construct the classifier or predictor efficiently; given large amount of data.
- **Interpretability :** It refers to what extent the classifier or predictor understands.

3.3 DECISION TREE INDUCTION

UQ. Define and explain: (i) Information Gain (ii) Entropy (iii) Gini Index **(MU - Dec. 2019)**

- Decision tree induction is the learning of decision trees from class-labeled training tuples.
- A decision tree is a structure that includes a root node, branches, and leaf nodes.

- Each internal node denotes a test on an attribute, each branch denotes the outcome of a test, and each leaf node holds a class label.
- The topmost node in the tree is the root node.
- The following decision tree is for the concept buys_laptop that indicates whether a customer at a company is likely to buy a computer or not. Each internal node represents a test on an attribute. Each leaf node represents a class (either buys_laptop = yes or buys_laptop = no)



(1C3)Fig. 3.3.1: Representation of a Decision Tree

- The benefits of having a decision tree are as follows:
  1. It does not require any domain knowledge.
  2. It is easy to comprehend.
  3. The learning and classification steps of a decision tree are simple and fast.

3.3.1 Decision Tree Induction Algorithm-

- A machine learning researcher named J. Ross Quinlan in 1980 developed a decision tree algorithm known as ID3 (Iterative Dichotomiser). Later, he presented C4.5, which was the successor of ID3. ID3 and C4.5 adopt a greedy approach.
- In this algorithm, there is no backtracking; the trees are constructed in a top-down recursive divide-and-conquer manner.

Algorithm : Generate_decision_tree. Generating a decision tree form training tuples of data partition D

Input

- Data partition, D, which is a set of training tuples and their associated class labels.
- attribute_list, the set of candidate attributes.
- Attribute_selection_method, a procedure to determine the splitting criterion that "best" partitions the data tuples into individual classes. This criterion includes a splitting_attribute and either a splitting point or splitting subset.

Output : A Decision Tree

Method

1. create a node N;
2. **if** tuples in D are all of the same class, C, **then**
3. return N as leaf node labeled with class C;
4. **if** attribute_list is empty **then**
5. return N as a leaf node with labeled with majority class in D; //majority voting
6. apply **attribute_selection_method**(D, attribute_list) to **find** the best splitting_criterion;
7. label node N with splitting_criterion;
8. **if** splitting_attribute is discrete-valued **and** multiway splits allowed **then**// no restricted to binary trees
9. attribute_list←attribute_list - splitting attribute; // remove splitting attribute
10. **for each** outcome j of splitting criterion
    // partition the tuples and grow subtrees for each partition
11. let D_j be the set of data tuples in D satisfying outcome j; // a partition
12. **if** D_j is empty **then**
13. attach a leaf labeled with the majority class in D to node N;
14. **else** attach the node returned by Generate_decision_tree (D_j, attribute list) to node N;
    end for
15. return N;

3.3.2 Tree Pruning

- The decision tree built may overfit the training data. There could be too many branches, some of which may reflect anomalies in the training data due to noise or outliers.
- Tree pruning addresses this issue of overfitting the data by removing the least reliable branches (using statistical measures).
- This generally results in a more compact and reliable decision tree that is faster and more accurate in its classification of data.

Module 3

- There are two approaches to prune a tree :
  - (i) **Pre-pruning :** The tree is pruned by halting its construction early.
  - (ii) **Post-pruning :** This approach removes a sub-tree from a fully grown tree.

Drawback of using a separate set of tuples to evaluate pruning

- If a separate set of tuples are used to evaluate pruning is that it may not be representative of the training tuples used to create the original decision tree.
- If the separate set of tuples are skewed, then using them to evaluate the pruned tree would not be a good indicator of the pruned tree's classification accuracy.
- Furthermore, using a separate set of tuples to evaluate pruning means there are less tuples to use for creation and testing of the tree. While this is considered a drawback in machine learning, it may not be so in data mining due to the availability of larger data sets.

GQ. Given a decision tree, you have the option of (a) converting the decision tree to rules and then pruning the resulting rules, or (b) pruning the decision tree and then converting the pruned tree to rules. What advantage does (a) have over (b)?

☑ **Ans. :**

If pruning a subtree, we would remove the subtree completely with method (b). However, with method (a), if pruning a rule, we may remove any precondition of it. The latter is less restrictive.

3.3.3 Cost Complexity

The cost complexity is measured by the following two parameters –

- Number of leaves in the tree, and
- Error rate of the tree

3.3.4 Classification using Information Gain (ID3)

- ID3 stands for Iterative Dichotomiser 3 and is named such because the algorithm iteratively (repeatedly) dichotomizes(divides) features into two or more groups at each step.
- Invented by Ross Quinlan, ID3 uses a **top-down greedy** approach to build a decision tree. In simple words, the **top-down** approach means that we start building the tree from the top and the **greedy** approach means that at each iteration we select the best feature at the present moment to create a node.
- Most generally ID3 is used for classification problems with nominal features only.

Metrics in ID3

As mentioned previously, the ID3 algorithm selects the best feature at each step while building a Decision tree.

- ID3 uses Information Gain or just Gain to find the best feature.
- **Information Gain** calculates the reduction in the entropy and measures how well a given feature separates or classifies the target classes. The feature with the highest Information Gain is selected as the best one.
- In simple words, **Entropy** is the measure of disorder and the Entropy of a dataset is the measure of disorder in the target feature of the dataset. Expected amount of information (in bits) needed to assign a class to a randomly drawn object is called Entropy.
- In the case of binary classification (where the target column has only two types of classes) entropy is 0 if all values in the target column are homogenous(similar) and will be 1 if the target column has equal number values for both the classes.
- We denote our dataset as D, entropy is calculated as :

$$\text{Entropy (D)} = \sum_{i=1}^{n} p_i \log_2 \frac{1}{p_i}$$

where,

n is the total number of classes in the target column

p_i is the **probability of class 'i'** or the ratio of *"number of rows with class i in the target column"* to the *"total number of rows"* in the dataset.

- Information Gain for a feature column A is calculated as :

$$\text{Gain(A)} = \text{Entropy(D)} - \text{Entropy(A)}$$

ID3 Steps

1. Calculate the Information Gain of each feature.
2. Considering that all rows don't belong to the same class, split the dataset D into subsets using the feature for which the Information Gain is maximum.
3. Make a decision tree node using the feature with the maximum Information gain.
4. If all rows belong to the same class, make the current node as a leaf node with the class as its label.
5. Repeat the above steps for the remaining features until we run out of all features, or the decision tree has all leaf nodes.

Ex. 3.3.1 : Apply ID3 algorithm on the following training dataset and extract the classification rule from the tree.

Day	Outlook	Temp.	Humidity	Wind	Play_Tennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

☑ **Soln. :**

Let the class label attributes be as follows:

C1 = Play_Tennis = Yes = 9 Samples

C2 = Play_Tennis = No = 5 Samples

Therefore, P(C1) = 9/14 and P(C2) = 5/14

(i) Entropy before split for the given database D:

$$H(D) = \sum_{i=1}^{s} p_i \log_2 \left(\frac{1}{p_i}\right) = \frac{9}{14}\log_2\frac{14}{9} + \frac{5}{14}\log_2\frac{14}{5}$$

$$= 0.4097 + 0.5305 = 0.940$$

(ii) Choosing Outlook as Splitting Attribute

Outlook	C1 Play_Tennis = Yes	C2 Play_Tennis = No	Entropy H
Sunny	2	3	0.971
Overcast	4	0	0
Rain	3	2	0.971

$$\therefore H(\text{outlook}) = \frac{5}{14} \times H(\text{Sunny}) + \frac{4}{14} \times H(\text{Overcast}) + \frac{5}{14} \times H(\text{Rain})$$

$$= \frac{5}{14} \times 0.971 + \frac{4}{14} \times 0 + \frac{5}{14} \times 0.971 = 0.694$$

$$\text{Gain (Outlook)} = H(D) - H(\text{Outlook}) = 0.940 - 0.694 = 0.246$$

(iii) Choosing Temperature as Splitting Attribute

Temperature	C1 Play_Tennis = Yes	C2 Play_Tennis = No	Entropy H
Hot	2	2	1
Mild	4	2	0.92
Cool	3	1	0.81

$$\therefore H(\text{Temperature}) = \frac{4}{14} \times H(\text{Hot}) + \frac{6}{14} \times H(\text{Mild}) + \frac{4}{14} \times H(\text{Cool})$$

$$= \frac{4}{14} \times 1 + \frac{6}{14} \times 0.92 + \frac{4}{14} \times 0.81 = 0.911$$

$$\text{Gain (Temperature)} = H(D) - H(\text{Temperature}) = 0.940 - 0.911 = 0.029$$

(iv) Choosing Humidity as Splitting Attribute

Humidity	C1 Play_Tennis = Yes	C2 Play_Tennis = No	Entropy H
High	3	4	0.985
Normal	6	1	0.592

$$\therefore H(\text{Humidity}) = \frac{7}{14} \times H(\text{High}) + \frac{7}{14} \times H(\text{Normal})$$

$$= \frac{7}{14} \times 0.985 + \frac{7}{14} \times 0.592 = 0.789$$

$$\text{Gain (Humidity)} = H(D) - H(\text{Humidity}) = 0.940 - 0.789 = 0.151$$

Module 3

(v) Choosing Wind as Splitting Attribute

Wind	C1 Play_Tennis = Yes	C2 Play_Tennis = No	Entropy H
Strong	3	3	1
Weak	6	2	0.811

$\therefore$ H(Wind) $= \frac{6}{14} \times$ H(Strong) $+ \frac{8}{14} \times$ H (Weak)

$= \frac{6}{14} \times 1 + \frac{8}{14} \times 0.811 = 0.892$

Gain (Wind) = H(D) – H(Wind)

= 0.940 – 0.892 = 0.048

Summary

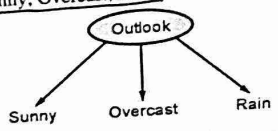
Gain(Outlook|D) = 0.246

Gain(Temperature|D) = 0.029

Gain(Humidity|D) = 0.151

Gain(Wind|D) = 0.048

Outlook attribute has the highest gain; therefore, it is used as the decision attribute in the root node. Since, Outlook has three possible values, the root node has three branches (Sunny, Overcast, Rain).



(IC4)**Fig. P. 3.3.1(a)**

Now, consider Outlook = Sunny and count the number of tuples from the original dataset D. Let us denote it as D1.

Day	Outlook	Temp.	Humidity	Wind	Play_Tennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

Let the class label attributes be as follows:

C1 = Play_Tennis = Yes|Sunny = 2 Samples

C2 = Play_Tennis = No|Sunny = 3 Samples

Therefore, P(C1) = 2/5 and P(C2) = 3/5

(i) Entropy before split for the given database D1 :

$H(D_1) = \sum_{i=1}^{s} p_i \log_2 \left(\frac{1}{p_i}\right) = \frac{2}{5}\log_2\frac{5}{2} + \frac{3}{5}\log_2\frac{5}{3} = 0.971$

(ii) Choosing Temperature as Splitting Attribute

Temperature	C1 Play_Tennis = Yes|Sunny	C2 Play_Tennis = No|Sunny	Entropy H
Hot	0	2	0
Mild	1	1	1
Cool	1	0	0

$\therefore$ H(Temperature) $= \frac{2}{5} \times$ H(Hot) $+ \frac{2}{5} \times$ H(Mild)

$+ \frac{1}{5} \times$ H(Cool)

$= \frac{2}{5} \times 0 + \frac{2}{5} \times 1 + \frac{1}{5} \times 0 = 0.4$

Gain (Temperature) = H(D1) – H(Temperature)

= 0.971 – 0.4 = 0.571

(iii) Choosing Humidity as Splitting Attribute

Humidity	C1 Play_Tennis = Yes|Sunny	C2 Play_Tennis = No|Sunny	Entropy H
High	0	3	0
Normal	2	0	0

H(Humidity) $= \frac{3}{5} \times$ H(High) $+ \frac{2}{5} \times$ H (Normal)

$= \frac{3}{5} \times 0 + \frac{2}{5} \times 0 = 0$

Gain (Humidity) = H(D1) – H(Humidity)

= 0.971 – 0 = 0.971

(iv) Choosing Wind as Splitting Attribute

Wind	C1 Play_Tennis = Yes|Sunny	C2 Play_Tennis = No|Sunny	Entropy H
Strong	1	1	1
Weak	1	2	0.918

$\therefore$ Gain (Wind) $= \frac{2}{5} \times$ H (Strong) $+ \frac{3}{5} \times$ H (Weak)

$= \frac{2}{5} \times 1 + \frac{3}{5} \times 0.918 = 0.951$

Gain (Wind) = H(D1) – H(Wind)

= 0.971 – 0.951 = 0.02

Summary

Gain(Temperature|D1) = 0.571

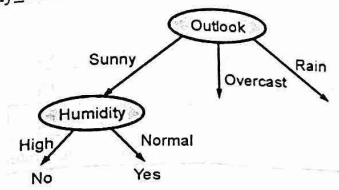
Gain(Humidity|D1) = 0.971

Gain(Wind|D1) = 0.02

Humidity attribute has the highest gain; therefore, it is placed below Outlook = "Sunny".

Since, Humidity has two possible values, the Humidity node has two branches (High, Normal).

From dataset D1, we find that when Humidity = High, Play_Tennis = No and when Humidity = Normal, Play_Tennis = Yes.

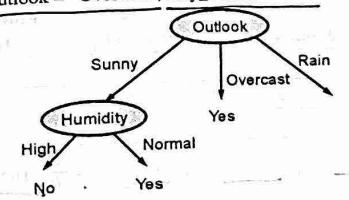


(IC5)**Fig. P. 3.3.1(b)**

- Now, consider Outlook = Overcast and count the number of tuples from the original dataset D. Let us denote it as D2.

Day	Outlook	Temp.	Humidity	Wind	Play_Tennis
3	Overcast	Hot	High	Weak	Yes
7	Overcast	Cool	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes

- From dataset D2, we find that for all values of Outlook = "Overcast", Play_Tennis = Yes.



(IC6) **Fig. P. 3.3.1(c)**

- Now, consider Outlook = Rain and count the number of tuples from the original dataset D. Let us denote it as D3.

Day	Outlook	Temp.	Humidity	Wind	Play_Tennis
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Let the class label attributes be as follows:

C1 = Play_Tennis = Yes|Rain = 3 Samples

C2 = Play_Tennis = No|Rain = 2 Samples

Therefore, P(C1) = 3/5 and P(C2) = 2/5

(i) Entropy before split for the given database D3:

$H(D3) = \sum_{i=1}^{s} p_i \log_2 \left(\frac{1}{p_i}\right)$

$\therefore$ H(D3) $= \frac{3}{5}\log_2\frac{5}{3} + \frac{2}{5}\log_2\frac{5}{2} = 0.971$

(ii) Choosing Temperature as Splitting Attribute

Temperature	C1 Play_Tennis = Yes|Rain	C2 Play_Tennis = No|Rain	Entropy H
Hot	0	0	0
Mild	2	1	0.918
Cool	1	1	1

$\therefore$ H (Temperature) $= \frac{0}{5} \times$ H(Hot) $+ \frac{3}{5} \times$ H(Mild)

$+ \frac{2}{5} \times$ H(Cool)

$= \frac{0}{5} \times 0 + \frac{3}{5} \times 0.918 + \frac{2}{5} \times 1 = 0.951$

Gain (Temperature) = H(D3) – H(Temperature)

= 0.971 – 0.951 = 0.02

(iii) Choosing Wind as Splitting Attribute

Wind	C1 Play_Tennis = Yes|Rain	C2 Play_Tennis = No|Rain	Entropy H
Strong	0	2	0
Weak	3	0	0

$\therefore$ H (Wind) $= \frac{2}{5} \times$ H (Strong) $+ \frac{3}{5} \times$ H (Weak)

$= \frac{2}{5} \times 0 + \frac{3}{5} \times 0 = 0$

Gain (Wind) = H(D3) – H(Wind)

= 0.971 – 0 = 0.971

Module 3

Summary

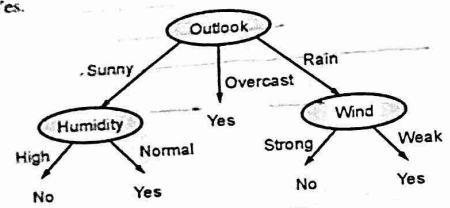
Gain(Temperature|D3) = 0.02

Gain(Wind|D3) = 0.971

Wind attribute has the highest gain; therefore, it is placed below Outlook = "Rain".

Since, Wind has two possible values, the Wind node has two branches (Strong, Weak).

From dataset D3, we find that when Wind = Strong, Play_Tennis = No and when Wind = Weak, Play_Tennis = Yes.



(1C7) **Fig. P. 3.3.1(d)**

The decision tree can also be expressed in rule format as :

IF Outlook = Sunny AND Humidity
= High THEN Play_Tennis = No

IF Outlook = Sunny AND Humidity
= Normal THEN Play_Tennis
= YES

IF Outlook = Overcast THEN Play_Tennis
= YES

IF Outlook = Rain AND Wind
= Strong THEN Play_Tennis
= No

IF Outlook = Rain AND Wind
= Weak THEN Play_Tennis
= YES

Ex. 3.3.2 : A simple example from the stock market involving only discrete ranges has profit as categorical attribute with values {Up, Down} and the training data is :

Age	Competition	Type	Profit
Old	Yes	Software	Down
Old	No	Software	Down
Old	No	Hardware	Down
Mid	Yes	Software	Down
Mid	Yes	Hardware	Down
Mid	No	Hardware	Up
Mid	No	Software	Up
New	Yes	Software	Up
New	No	Hardware	Up
New	No	Software	Up

Apply decision tree algorithm and show the generated rules.

☑ **Soln. :**

Let the class label attributes be as follows:

C1 = Profit = Down = 5 Samples

C2 = Profit = Up = 5 Samples

Therefore, P(C1) = 5/10 and P(C2) = 5/10

(i) Entropy before split for the given database D :

$$H(D) = \sum_{i=1}^{s} p_i \log_2\left(\frac{1}{p_i}\right)$$

$$\therefore H(D) = \frac{5}{10}\log_2\frac{10}{5} + \frac{5}{10}\log_2\frac{10}{5} = 0.5 + 0.5 = 1$$

(ii) Choosing Age as the Splitting Attribute.

Age	C1 Profit = Down	C2 Profit = Up	Entropy H
Old	3	0	0
Mid	2	2	1
New	0	3	0

$$\therefore H(Age) = \frac{3}{10}\times H(Old) + \frac{4}{10}\times H(Mild) + \frac{3}{10}\times H(New)$$

$$= \frac{3}{10}\times 0 + \frac{4}{10}\times 1 + \frac{3}{10}\times 0 = 0.4$$

$$Gain(Age) = H(D) - H(Age) = 1 - 0.4 = 0.6$$

(iii) Choosing Competition as the Splitting Attribute.

Competition	C1 Profit = Down	C2 Profit = Up	Entropy H
Yes	3	1	0.8113
No	2	4	0.9183

$$\therefore H(Competition) = \frac{4}{10}\times H(Yes) + \frac{6}{10}\times H(No)$$

$$= \frac{4}{10}\times 0.8113 + \frac{6}{10}\times 0.9183 = 0.8755$$

$$Gain(Competition) = H(D) - H(Competition)$$

$$= 1 - 0.8755 = 0.1245$$

(iv) Choosing Type as the Splitting Attribute.

Type	C1 Profit = Down	C2 Profit = Up	Entropy H
Software	3	3	1
Hardware	2	2	1

$$H(Type) = \frac{6}{10}\times H(Software) + \frac{4}{10}\times H(Hardware)$$

$$= \frac{6}{10}\times 1 + \frac{4}{10}\times 1 = 1$$

$$Gain(Type) = H(D) - H(Type) = 1 - 1 = 0$$

Summary

Gain(Age) = 0.6

Gain(Competition) = 0.1245

Gain (Type) = 0

Age attribute has the highest gain; therefore, it is used as the decision attribute in the root node.

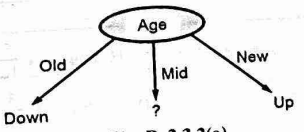
Since Age has three possible values, the root node has three branches (Old, Mid, New).

From dataset we find that,

IF Age = Old THEN Profit = Down

IF Age = Mid THEN Profit = Down OR Profit = Up

If Age = New THEN Profit = Up



(1C8) **Fig. P. 3.3.2(a)**

Now, consider Age = Mid and count the number of tuples from the original dataset D. Let us denote it as D1.

Age	Competition	Type	Profit
Mid	Yes	Software	Down
Mid	Yes	Hardware	Down
Mid	No	Hardware	Up
Mid	No	Software	Up

Let the class label attributes be as follows:

C1 = Profit = Down = 2 Samples

C2 = Profit = Up = 2 Samples

Therefore, P(C1) = 2/4 and P(C2) = 2/4

(i) Entropy before split for the given database D1:

$$H(D1) = \sum_{i=1}^{s} p_i \log_2\left(\frac{1}{p_i}\right)$$

$$\therefore H(D1) = \frac{2}{4}\log_2\frac{4}{2} + \frac{2}{4}\log_2\frac{4}{2} = 0.5 + 0.5 = 1$$

(ii) Choosing Competition as the Splitting Attribute.

Competition	C1 Profit = Down	C2 Profit = Up	Entropy H
Yes	2	0	0
No	0	2	0

$$\therefore H(Competition) = \frac{2}{4}\times H(Yes) + \frac{2}{4}\times H(No)$$

$$= \frac{2}{4}\times 0 + \frac{2}{4}\times 0 = 0$$

$$Gain(Competition) = H(D1) - H(Competition)$$

$$= 1 - 0 = 1$$

(iii) Choosing Type as the Splitting Attribute.

Type	C1 Profit = Down	C2 Profit = Up	Entropy H
Software	1	1	1
Hardware	1	1	1

$$\therefore H(Type) = \frac{2}{4}\times H(Software) + \frac{2}{4}\times H(Hardware)$$

$$= \frac{2}{4}\times 1 + \frac{2}{4}\times 1 = 1$$

$$Gain(Competition) = H(D1) - H(Competition)$$

$$= 1 - 0 = 1$$

$$Gain(Type) = H(D) - H(Type) = 1 - 1 = 0$$

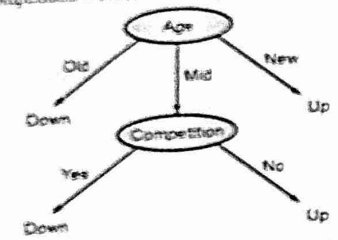
Module 3

Summary

Gain(CompetitionID1) = 1

Gain (TypeID1) = 0

- **Competition** attribute has the highest gain, therefore, it is placed below Age = "Mid"
- Since, Competition has two possible values, the Competition node has two branches (Yes, No).
- From dataset D1, we find that when Competition = Yes, Profit = Down and when Competition = No, Profit = Up.



(1C9) **Fig. P. 3.3.2(a)**

The decision tree can also be expressed in rule format as:

IF Age = Old THEN Profit = Down

IF Age = Mid AND Competition = Yes THEN Profit = Down

IF Age = Mid AND Competition = No THEN Profit = Up

If Age = New THEN Profit = Up

Ex. 3.3.3 : Using the following training data set, create classification model using decision tree and draw the final tree.

Tid	Income	Age	Own House
1	Very High	Young	Yes
2	High	Medium	Yes
3	Low	Young	Rented
4	High	Medium	Yes
5	Very High	Medium	Yes
6	Medium	Young	Yes
7	High	Old	Yes
8	Medium	Medium	Rented
9	Low	Medium	Rented
10	Low	Old	Rented
11	High	Young	Yes
12	Medium	Old	Rented

Soln. :

Let the class label attributes be as follows:

C1 = Own House = Yes = 7 Samples

C2 = Own House = Rented = 5 Samples

Therefore, P(C1) = 7/12 and P(C2) = 5/12

(i) Entropy before split for the given database D:

$$H(D) = \sum_{i=1}^{2} p_i \log_2 \left(\frac{1}{p_i}\right)$$

$$\therefore H(D) = \frac{7}{12}\log_2\frac{12}{7} + \frac{5}{12}\log_2\frac{12}{5} = 0.454 + 0.526 = 0.980$$

(ii) Choosing Income as the Splitting Attribute

Income	C1 Own House = Yes	C2 Own House = Rented	Entropy H
Very High	2	0	0
High	4	0	0
Medium	1	2	0.918
Low	0	3	0

$$\therefore H(\text{Income}) = \frac{2}{12} \times H(\text{Very High}) + \frac{4}{12} \times H(\text{High}) + \frac{3}{12} \times H(\text{Medium}) + \frac{3}{12} \times H(\text{Low})$$

$$= \frac{2}{12}\times 0 + \frac{4}{12}\times 0 + \frac{3}{12}\times 0.918 + \frac{3}{12}\times 0 = 0.229$$

$$\text{Gain(Income)} = H(D) - H(\text{Income}) = 0.980 - 0.229 = 0.751$$

(iii) Choosing Age as the Splitting Attribute

Age	C1 Own House = Yes	C2 Own House = Rented	Entropy H
Young	3	1	0.811
Medium	3	2	0.971
Old	1	2	0.918

$$\therefore H(\text{Age}) = \frac{4}{12} \times H(\text{Young}) + \frac{5}{12} \times H(\text{Medium}) + \frac{3}{12} \times H(\text{Old})$$

$$= \frac{4}{12}\times 0.811 + \frac{5}{12}\times 0.971 + \frac{3}{12}\times 0.918 = 0.904$$

$$\text{Gain(Age)} = H(D) - H(\text{Age}) = 0.980 - 0.904 = 0.076$$

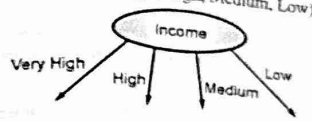
Summary:

Gain(Income) = 0.751

Gain(Age) = 0.076

Income attribute has the highest gain; therefore, it is used as the decision attribute in the root node.

Since Income has four possible values, the root node has four branches (Very High, High, Medium, Low).

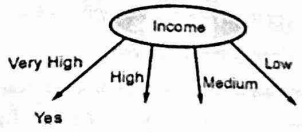


(1C10) **Fig. P. 3.3.3(a)**

Now, consider Income = Very High and count the number of tuples from the original dataset D. Let us denote it as D1.

Tid	Income	Age	Own House
1	Very High	Young	Yes
5	Very High	Medium	Yes

Since both the tuples have class label Own House = Yes, we directly give "Yes" as a class below Income = Very High.

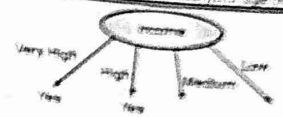


(1C11) **Fig. P. 3.3.3(b)**

Now, consider Income = High and count the number of tuples from the original dataset D. Let us denote it as D2.

Tid	Income	Age	Own House
2	High	Medium	Yes
4	High	Medium	Yes
7	High	Old	Yes
11	High	Young	Yes

Since all the four tuples have class label Own House = Yes, we directly give "Yes" as a class below Income = High.

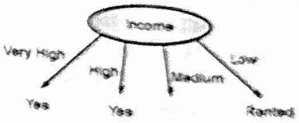


(1C12) **Fig. P. 3.3.3(c)**

Now, consider Income = Low and count the number of tuples from the original dataset D. Let us denote it as D3.

Tid	Income	Age	Own House
3	Low	Young	Rented
9	Low	Medium	Rented
10	Low	Old	Rented

Since all the three tuples have class label Own House = Rented, we directly give "Rented" as a class below Income = Low.



(1C13) **Fig. P. 3.3.3(d)**

Now, consider Income = Medium and count the number of tuples from the original dataset D. Let us denote it as D4.

Tid	Income	Age	Own House
6	Medium	Young	Yes
8	Medium	Medium	Rented
12	Medium	Old	Rented

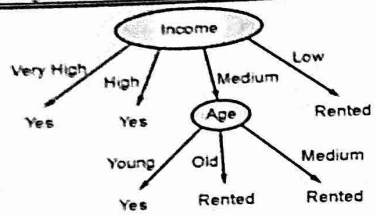
We find that,

IF Income = Medium AND Age = Young THEN Own House = Yes

IF Income = Medium AND Age = Medium THEN Own House = Rented

IF Income = Medium AND Age = Old THEN Own House = Rented

Module 3



(IC14) **Fig. P. 3.3.3(e)**

The decision tree can also be expressed in rule format as:

IF Income = Very High THEN Own House = Yes
IF Income = High THEN Own House = Yes
IF Income = Low THEN Own House = Rented
IF Income = Medium AND Age
= Young THEN Own House = Yes
IF Income = Medium AND Age
= Medium THEN Own House
= Rented
IF Income = Medium AND Age
= Old THEN Own House = Rented

Ex. 3.3.4 : The following table consists of training data from an employee database. The data have been generalized. For example, "31 ... 35" for age represents the age range of 31 to 35. For a given row entry, count represents the number of data tuples having the values for department, status, age, and salary given in that row.

department	status	Age	Salary	count
Sales	senior	31...35	46K...50K	30
Sales	junior	26...30	26K...30K	40
Sales	junior	31...35	31K...35K	40
Systems	junior	21...25	46K...50K	20
Systems	senior	31...35	66K...70K	5
Systems	junior	26...30	46K...50K	3
Systems	senior	41...45	66K...70K	3
marketing	senior	36...40	46K...50K	10
marketing	junior	31...35	41K...45K	4
Secretary	senior	46...50	36K...40K	4
Secretary	junior	26...30	26K...30K	6

Let status be the class label attribute.

(a) How would you modify the basic decision tree algorithm to take into consideration the count of each generalized data tuple (i.e., of each row entry)?

(b) Use your algorithm to construct a decision tree from the given data.

☑ **Soln. :**

(a) The basic decision tree algorithm should be modified as follows to take into consideration the count of each generalized data tuple.

- The count of each tuple must be integrated into the calculation of the attribute selection measure (such as information gain).
- Take the count into consideration to determine the most common class among the tuples.

(b) Use **ID3 algorithm** to construct a decision tree from the given data.

Let the class label attributes be as follows:

C1 = status = junior = 113 Samples

C2 = status = senior = 52 Samples

Therefore, P(C1) = 113/165 and P(C2) = 52/165

(i) Entropy before split for the given database D:

$$H(D) = \sum_{i=1}^{s} p_i \log_2\left(\frac{1}{p_i}\right)$$

$$\therefore H(D) = \frac{113}{165}\log_2\frac{165}{113} + \frac{52}{165}\log_2\frac{165}{52}$$

$$= 0.3740 + 0.5250 = 0.899$$

(ii) Choosing department at the Splitting Attribute.

Department	C1 Status = junior	C2 Status = senior	Entropy H
Sales	80	30	0.8454
Systems	23	8	0.8238
Marketing	4	10	0.8631
Secretary	6	4	0.9709

$$\therefore H(\text{department}) = \frac{110}{165} \times H(\text{Sales}) + \frac{31}{165} \times H(\text{systems}) + \frac{14}{165} \times H(\text{marketing}) + \frac{10}{165} \times H(\text{secretary})$$

$$= \frac{110}{165} \times 0.8454 + \frac{31}{165} \times 0.8238 + \frac{14}{165} \times 0.8631 + \frac{10}{165} \times 0.9709 = 0.8504$$

Gain(department) = H(D) – H(department)
= 0.899 – 0.8504 = 0.0486

(iii) Choosing age as the Splitting Attribute.

age	C1 Status=junior	C2 Status=senior	Entropy H
21...25	20	0	0
26...30	49	0	0
31...35	44	35	0.9906
36...40	0	10	0
41...45	0	3	0
46...50	0	4	0

$$\therefore H(\text{age}) = \frac{20}{165} \times H(21....25) + \frac{49}{165} \times H(26.....30) + \frac{79}{165} \times H(31.....35) + \frac{10}{165} \times H(36.....40) + \frac{3}{165} \times H(41.....45) + \frac{4}{165} \times H(46....50)$$

$$= \frac{20}{165} \times 0 + \frac{49}{165} \times 0 + \frac{79}{165} \times 0.9906 + \frac{10}{165} \times 0 + \frac{3}{165} \times 0 + \frac{4}{165} \times 0 = 0.4743$$

Gain(age) = H(D) – H(age)
= 0.899 – 0.4743 = 0.4247

(iv) Choosing salary as the Splitting Attribute.

Salary	C1 Status=junior	C2 Status=senior	Entropy H
26K...30K	46	0	0
31K...35K	40	0	0
36K...40K	0	4	0
41K...45K	4	0	0
46K...50K	23	40	0.9468
66K...70K	0	8	0

$$\therefore H(\text{salary}) = \frac{46}{165} \times H(26K....30K) + \frac{40}{165} \times H(31K....35K) + \frac{4}{165} \times H(36K....40K) + \frac{4}{165} \times H(41K....45K) + \frac{63}{165} \times H(46K....50K) + \frac{8}{165} \times H(66K....70K)$$

$$= \frac{46}{165} \times 0 + \frac{40}{165} \times 0 + \frac{4}{165} \times 0 + \frac{4}{165} \times 0 + \frac{63}{165} \times 0.9468 + \frac{8}{165} \times 0 = 0.3615$$

Gain (salary) = H(D) – H(salary)
= 0.899 – 0.3615 = 0.5375

Summary

Gain(department) = 0.0486
Gain(age) = 0.4247
Gain(salary) = 0.5375

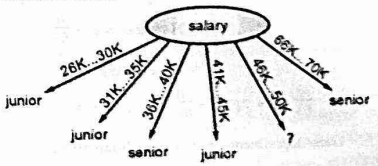
salary attribute has the highest gain; therefore, it is used as the decision attribute in the root node.

Since salary has six possible values, the root node has six branches (26K...30K, 31K...35K, 36K...40K, 41K...45K, 46K...50K, 66K...70K).

From the dataset D, we find that

IF salary = 26K...30K THEN status = junior
IF salary = 30K...35K THEN status = junior
IF salary = 36K...40K THEN status = senior
IF salary = 41K...45K THEN status = junior
IF salary = 46K...50K THEN status = junior OR status = senior
IF salary = 66K...70K THEN status = senior

Only one branch i.e. 46K...50K is not giving a unique class label attribute.



(IC15) **Fig. P. 3.3.4(a)**

Module 3

Now, consider salary = 46K...50K and count the number of tuples from the original dataset D. Let us denote it as D1.

department	status	Age	Salary	count
Sales	senior	31...35	46K...50K	30
Systems	junior	21...25	46K...50K	20
Systems	junior	26...30	46K...50K	3
marketing	senior	36...40	46K...50K	10

Let the class label attributes be as follows:

C1 = status = junior|salary = 46K...50K

= 23 Samples

C2 = status = senior|salary = 46K...50K

= 40 Samples

Therefore, P(C1) = 23/63 and P(C2) = 40/63

(i) Entropy before split for the given database D1:

$$H(D1) = \sum_{i=1}^{s} p_i \log_2\left(\frac{1}{p_i}\right)$$

$$\therefore H(D1) = \frac{23}{63}\log_2\frac{63}{23} + \frac{40}{63}\log_2\frac{63}{40} = 0.9468$$

(ii) Choosing department as the Splitting Attribute.

department	C1 Status=junior	C2 Status=senior	Entropy H
Sales	0	30	0
Systems	23	0	0
Marketing	0	10	0
Secretary	0	0	0

$$\therefore H(\text{department}) = \frac{30}{63}\times H(\text{sales}) + \frac{23}{63} \times H(\text{systems}) + \frac{10}{63} \times H(\text{marketing}) + \frac{0}{63} \times H(\text{secretary})$$

$$= \frac{30}{63}\times 0 + \frac{23}{63}\times 0 + \frac{10}{63}\times 0 + \frac{0}{63}\times 0 = 0$$

Gain(department) = H(D1) – H(department)

= 0.9468 – 0 = 0.9468

(iii) Choosing age as the Splitting Attribute.

age	C1 Status=junior	C2 Status=senior	Entropy H
21...25	20	0	0
26...30	3	0	0
31...35	0	30	0
36...40	0	10	0
41...45	0	0	0
46...50	0	0	0

$$\therefore H(\text{age}) = \frac{20}{63}\times H(21...25) + \frac{3}{63}\times H(26...30) + \frac{30}{63}\times H(31....35) + \frac{10}{63}\times H(36.....40) + \frac{0}{63}\times H(41.....45) + \frac{0}{63}\times H(46.....50)$$

$$= \frac{20}{63}\times 0 + \frac{3}{63}\times 0 + \frac{30}{63}\times 0 + \frac{10}{63}\times 0 + \frac{0}{63}\times 0 + \frac{0}{63}\times 0 = 0$$

Gain(age) = H(D1) – H(age)

= 0.9468 – 0 = 0.9468

Summary

Gain(department|D1) = 0.9468

Gain(age|D1) = 0.9468

Both the attributes have the same gain; therefore, we choose one of the attribute arbitrarily and place it below "salary = 46K...50K".

We choose **department** as the attribute below "salary = 46K...50K".

From dataset D1, we find that

IF salary = 46K...50K AND department = sales

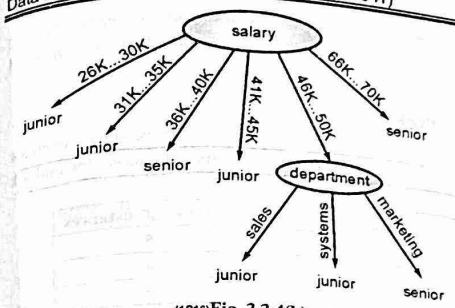
THEN status = junior.

IF salary = 46K...50K AND department = systems

THEN status = junior.

IF salary = 46K...50K AND department = marketing

THEN status = senior.



(1C16)Fig. 3.3.4(b)

The decision tree can also be expressed in rule format as:

IF salary = 26K...30K THEN status = junior

IF salary = 30K...35K THEN status = junior

IF salary = 36K...40K THEN status = senior

IF salary = 41K...45K THEN status = junior

IF salary = 46K...50K AND department = sales THEN status = junior

IF salary = 46K...50K AND department = systems THEN status = junior

IF salary = 46K...50K AND department = marketing THEN status = senior

IF salary = 66K...70K THEN status = senior

Advantages of ID3 Algorithm

1. Evident prediction rules are constructed form the training data.
2. It builds the short tree.
3. It searches entire dataset to create the tree.
4. It searches complete hypothesis space to predict unlabeled instances.
5. It is less sensitive toward errors of individual training examples because of statistical properties of instances are utilized

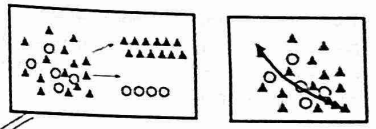
Disadvantages of ID3 Algorithm

1. Over-fitting of the data may happen while classification.
2. It does not perform backtracking while searching.
3. It may converge in locally optimal solution.
4. Computational complexity may be very high for the continuous data.

3.3.5 Classification using CART Algorithm

- CART (Classification and Regression Tree) is an alternative decision tree building algorithm.
- It can handle both classification and regression tasks.
- This algorithm uses a new metric named **gini index** to create decision points for classification tasks.
- The CART or Classification & Regression Trees methodology was introduced in 1984 by Leo Breiman, Jerome Friedman, Richard Olshen and Charles Stone as an umbrella term to refer to the following types of decision trees:

1. **Classification Trees :** In these trees, the target variable is categorical and the tree is used to identify the "class" within which a target variable would likely fall into.



(a) Classification Tree (b) Regression Tree

Fig. 3.3.2 : Representation of a CART Decision Tree

2. **Regression Trees :** In these trees, the target variable is continuous and tree is used to predict its value.

- It is characterized by the fact that it constructs binary trees, namely each internal node has exactly two outgoing edges.
- When provided, CART can consider misclassification costs in the tree induction. It also enables users to provide prior probability distribution.
- An important feature of CART is its ability to generate regression trees. Regression trees are trees where their leaves predict a real number and not a class. In case of regression, CART looks for splits that minimize the prediction squared error (the least-squared deviation). The prediction in each leaf is based on the weighted mean for node.

Module 3

- **Gini index :** Gini index is a metric for classification tasks in CART. It stores sum of squared probabilities of each class. We can formulate it as illustrated below.

Gini (Attribute = value) = $1 - \Sigma (P_i)^2$ for i=1 to number of classes

Gini (Attribute) = $\Sigma_{v=values} P_v \times$ Gini (Attribute = value)

Note : Select the attribute as the node of decision tree whose Gini Index is Low.

Advantages of CART

1. Decision trees can inherently perform multiclass classification.
2. They provide most model interpretability because they are simply series of if-else conditions.
3. .They can handle both numerical and categorical data.
4. Nonlinear relationships among features do not affect the performance of the decision trees.

Disadvantages of CART

1. A small change in the dataset can make the tree structure unstable which can cause variance.
2. Decision tree learners create underfit trees if some classes are imbalanced. It is therefore recommended to balance the data set prior to fitting with the decision tree.

Ex. 3.3.5 : Apply CART algorithm on the following training dataset and construct the decision tree.

Day	Outlook	Temperature	Humidity	Wind	Play_Tennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

Soln. : There are 14 instances of tennis playing decisions based on outlook, temperature, humidity and wind factors.

Outlook

Outlook is a nominal feature. It can be sunny, overcast or rain. We will summarize the final decisions for outlook feature.

Outlook	Yes	No	Number of instances
Sunny	2	3	5
Overcast	4	0	4
Rain	3	2	5

Gini(Outlook=Sunny) = $1 - (2/5)^2 - (3/5)^2$

= 1 – 0.16 – 0.36 = 0.48

Gini(Outlook=Overcast) = $1 - (4/4)^2 - (0/4)^2 = 0$

Gini(Outlook=Rain) = $1 - (3/5)^2 - (2/5)^2$

= 1 – 0.36 – 0.16 = 0.48

Then, we will calculate weighted sum of gini indexes for outlook feature.

Gini(Outlook) = (5/14) × 0.48 + (4/14) × 0 + (5/14) × 0.48

= 0.171 + 0 + 0.171 = 0.342

Temperature

Similarly, temperature is a nominal feature and it could have 3 different values : Cool, Hot and Mild. Let's summarize decisions for temperature feature.

Temperature	Yes	No	Number of instances
Hot	2	2	4
Cool	3	1	4
Mild	4	2	6

Gini(Temp=Hot) = $1 - (2/4)^2 - (2/4)^2 = 0.5$

Gini(Temp=Cool) = $1 - (3/4)^2 - (1/4)^2$

= 1 – 0.5625 – 0.0625 = 0.375

Gini(Temp=Mild) = $1 - (4/6)^2 - (2/6)^2$

= 1 – 0.444 – 0.111 = 0.445

We'll calculate weighted sum of gini index for temperature feature

Gini(Temperature) = (4/14) × 0.5 + (4/14) × 0.375

+ (6/14) × 0.445 = 0.142 + 0.107 + 0.190 = 0.439

Humidity

Humidity is a binary class feature. It can be high or normal.

Humidity	Yes	No	Number of instances
High	3	4	7
Normal	6	1	7

Gini(Humidity=High) = $1 - (3/7)^2 - (4/7)^2$

= 1 – 0.183 – 0.326 = 0.489

Gini(Humidity=Normal) = $1 - (6/7)^2 - (1/7)^2$

= 1 – 0.734 – 0.02 = 0.244

Weighted sum for humidity feature will be calculated next

Gini(Humidity) = (7/14) x 0.489 + (7/14) x 0.244 = 0.367

Wind

Wind is a binary class similar to humidity. It can be weak and strong.

Wind	Yes	No	Number of instances
Weak	6	2	8
Strong	3	3	6

Gini(Wind=Weak) = $1 - (6/8)^2 - (2/8)^2$

= 1 – 0.5625 – 0.062 = 0.375

Gini(Wind=Strong) = $1 - (3/6)^2 - (3/6)^2$

= 1 – 0.25 – 0.25 = 0.5

Weighted sum for wind feature will be calculated next

Gini(Wind) = (8/14) x 0.375 + (6/14) x 0.5 = 0.428

Time to decide

We've calculated gini index values for each feature. The winner will be outlook feature because its cost is the lowest.

Feature	Gini index
Outlook	0.342
Temperature	0.439
Humidity	0.367
Wind	0.428

We'll put **outlook** decision at the top of the tree.

Fig. P. 3.3.5(a)

Now, consider Outlook = Sunny and count the number of tuples from the original dataset D. Let us denote it as D1.

Day	Outlook	Temp.	Humidity	Wind	Play_Tennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

Gini of temperature for sunny outlook

Temperature	Yes	No	Number of instances
Hot	0	2	2
Cool	1	0	1
Mild	1	1	2

Gini(Outlook=Sunny and Temp.=Hot)

= $1 - (0/2)^2 - (2/2)^2 = 0$

Gini(Outlook=Sunny and Temp.=Cool)

= $1 - (1/1)^2 - (0/1)^2 = 0$

Gini(Outlook=Sunny and Temp.=Mild)

= $1 - (1/2)^2 - (1/2)^2 = 1 - 0.25 - 0.25 = 0.5$

Gini(Outlook=Sunny and Temp.)

= (2/5) × 0 + (1/5) × 0 + (2/5) ×0.5 = 0.2

Gini of humidity for sunny outlook

Humidity	Yes	No	Number of instances
High	0	3	3
Normal	2	0	2

Gini(Outlook=Sunny and Humidity=High)

= $1 - (0/3)^2 - (3/3)^2 = 0$

Gini(Outlook=Sunny and Humidity=Normal)

= $1 - (2/2)^2 - (0/2)^2 = 0$

Gini(Outlook=Sunny and Humidity)

= (3/5) × 0 + (2/5) × 0 = 0

Gini of wind for sunny outlook

Wind	Yes	No	Number of instances
Weak	1	2	3
Strong	1	1	2

Gini(Outlook=Sunny and Wind=Weak)

= $1 - (1/3)^2 - (2/3)^2 = 0.266$

Gini(Outlook=Sunny and Wind=Strong)

= $1 - (1/2)^2 - (1/2)^2 = 0.2$

Module 3

Gini(Outlook=Sunny and Wind)
$= (3/5) \times 0.266 + (2/5) \times 0.2 = 0.466$

Decision for sunny outlook

We've calculated gini index scores for feature when outlook is sunny. **The winner is humidity because it has the lowest value.**

Feature	Gini index
Temperature	0.2
Humidity	0
Wind	0.466

Since, Humidity has two possible values, the Humidity node has two branches (High, Normal).

From dataset D1, we find that when Humidity = High, Play_Tennis = No and when Humidity = Normal, Play_Tennis = Yes.

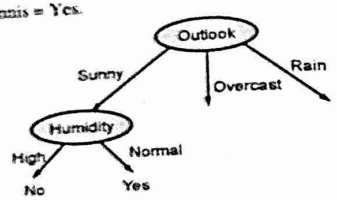


Fig. P. 3.3.5(b)

Now, consider Outlook = Overcast and count the number of tuples from the original dataset D. Let us denote it as D2.

Day	Outlook	Temp.	Humidity	Wind	Play_Tennis
3	Overcast	Hot	High	Weak	Yes
7	Overcast	Cool	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes

From dataset D2, we find that for all values of Outlook = "Overcast", Play_Tennis = Yes.

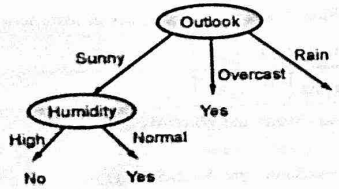


Fig. P. 3.3.5(c)

Now, consider Outlook = Rain and count the number of tuples from the original dataset D. Let us denote it as D3.

Day	Outlook	Temp.	Humidity	Wind	Play_Tennis
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
10	Rain	Mild	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

We'll calculate gini index scores for temperature, humidity and wind features when outlook is rain.

Gini of temperature for rain outlook

Temperature	Yes	No	Number of instances
Cool	1	1	2
Mild	2	1	3

Gini(Outlook=Rain and Temp.=Cool)
$= 1 - (1/2)^2 - (1/2)^2 = 0.5$

Gini(Outlook=Rain and Temp.=Mild)
$= 1 - (2/3)^2 - (1/3)^2 = 0.444$

Gini(Outlook=Rain and Temp.)
$= (2/5) \times 0.5 + (3/5) \times 0.444 = 0.466$

Gini of humidity for rain outlook

Humidity	Yes	No	Number of instances
High	1	1	2
Normal	2	1	3

Gini(Outlook=Rain and Humidity=High)
$= 1 - (1/2)^2 - (1/2)^2 = 0.5$

Gini(Outlook=Rain and Humidity=Normal)
$= 1 - (2/3)^2 - (1/3)^2 = 0.444$

Gini(Outlook=Rain and Humidity)
$= (2/5) \times 0.5 + (3/5) \times 0.444 = 0.466$

Gini of wind for rain outlook

Wind	Yes	No	Number of instances
Weak	3	0	3
Strong	0	2	2

Gini(Outlook=Rain and Wind=Weak)
$= 1 - (3/3)^2 - (0/3)^2 = 0$

Gini(Outlook=Rain and Wind=Strong)
$= 1 - (0/2)^2 - (2/2)^2 = 0$

Gini(Outlook=Rain and Wind) $= (3/5) \times 0 + (2/5) \times 0 = 0$

Decision for rain outlook

The winner is wind feature for rain outlook because it has the minimum gini index score in features.

Feature	Gini index
Temperature	0.466
Humidity	0.466
Wind	0

Put the wind feature for rain outlook branch and monitor the new sub data sets.

Since, Wind has two possible values, the Wind node has two branches (Strong, Weak).

From dataset D3, we find that when Wind = Strong, Play_Tennis = No and when Wind = Weak, Play_Tennis = Yes.

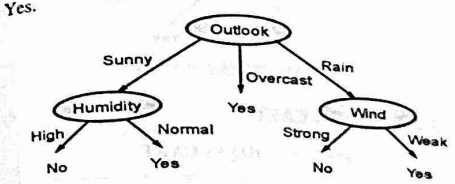


Fig. P. 3.3.5(d)

The decision tree can also be expressed in rule format as:

IF Outlook = Sunny AND Humidity = High
THEN Play_Tennis = No
IF Outlook = Sunny AND Humidity = Normal
THEN Play_Tennis = YES
IF Outlook = Overcast THEN Play_Tennis = YES
IF Outlook = Rain AND Wind = Strong
THEN Play_Tennis = No
IF Outlook = Rain AND Wind = Weak
THEN Play_Tennis = YES

Ex. 3.3.6 : For the following dataset of Employee Selection given below, compute gini index of each attribute and determine which attribute is the root attribute.

Tuples	Age	Salary	Job	Performance	Select
1	Young	High	Private	Average	No
2	Young	High	Private	Excellent	No
3	Middle-aged	High	Private	Average	Yes
4	Old	Medium	Private	Average	Yes
5	Old	Low	Government	Average	Yes
6	Old	Low	Government	Excellent	No
7	Middle-aged	Low	Government	Excellent	Yes
8	Young	Medium	Private	Average	No
9	Young	Low	Government	Average	Yes
10	Old	Medium	Government	Average	Yes
11	Young	Medium	Government	Excellent	Yes
12	Middle-aged	Medium	Private	Excellent	Yes
13	Middle-aged	High	Government	Average	Yes
14	Old	Medium	Private	Excellent	No

Module 3

☑ **Soln. :**

There are 14 instances of selection decisions based on age, salary, job and performance factors.

Age

Age is a nominal feature. It can be young, old or middle-aged. We will summarize the final decisions for age feature.

Age	Yes	No	Number of instances
Young	2	3	5
Middle-Aged	4	0	4
Old	3	2	5

Gini(Age=Young) $= 1 - (2/5)^2 - (3/5)^2$
$= 1 - 0.16 - 0.36 = 0.48$

Gini(Age = Middle-aged) $= 1 - (4/4)^2 - (0/4)^2 = 0$

Gini(Outlook=Old) $= 1 - (3/5)^2 - (2/5)^2$
$= 1 - 0.36 - 0.16 = 0.48$

Then, we will calculate weighted sum of gini indexes for age feature.

Gini(Age) $= (5/14) \times 0.48 + (4/14) \times 0 + (5/14) \times 0.48$
$= 0.171 + 0 + 0.171 = 0.342$

Salary

Similarly, salary is a nominal feature and it could have 3 different values: High, Medium and Low. Let's summarize decisions for salary feature.

Salary	Yes	No	Number of instances
High	2	2	4
Low	3	1	4
Medium	4	2	6

$Gini(Salary=High) = 1 - (2/4)^2 - (2/4)^2 = 0.5$

$Gini(Salary=Low) = 1 - (3/4)^2 - (1/4)^2$

$= 1 - 0.5625 - 0.0625 = 0.375$

$Gini(Salary=Medium) = 1 - (4/6)^2 - (2/6)^2$

$= 1 - 0.444 - 0.111 = 0.445$

We'll calculate weighted sum of gini index for salary feature

$Gini(Salary) = (4/14) \times 0.5 + (4/14) \times 0.375 + (6/14) \times 0.445$

$= 0.142 + 0.107 + 0.190 = 0.439$

Job

Humidity is a binary class feature. It can be private or government.

Job	Yes	No	Number of instances
Private	3	4	7
Govt.	6	1	7

$Gini(Job=Private) = 1 - (3/7)^2 - (4/7)^2$

$= 1 - 0.183 - 0.326 = 0.489$

$Gini(Job=Govt.) = 1 - (6/7)^2 - (1/7)^2$

$= 1 - 0.734 - 0.02 = 0.244$

Weighted sum for job feature will be calculated next

$Gini(Job) = (7/14) \times 0.489 + (7/14) \times 0.244 = 0.367$

Performance

Performance is a binary class similar to job. It can be average or excellent.

Performance	Yes	No	Number of instances
Average	6	2	8
Excellent	3	3	6

$Gini(Performance=Average) = 1 - (6/8)^2 - (2/8)^2$

$= 1 - 0.5625 - 0.062 = 0.375$

$Gini(Performance=Excellent) = 1 - (3/6)^2 - (3/6)^2$

$= 1 - 0.25 - 0.25 = 0.5$

Weighted sum for performance feature will be calculated next

$Gini(Performance) = (8/14) \times 0.375 + (6/14) \times 0.5 = 0.428$

Time to decide

We've calculated gini index values for each feature. The winner will be age feature because its cost is the lowest.

Feature	Gini index
Age	0.342
Salary	0.439
Job	0.367
Performance	0.428

We'll put age decision at the top of the tree. From the dataset it is clear that for age = middle-aged, select = yes.

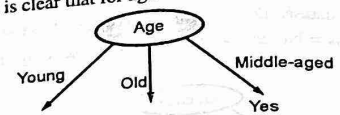


Fig. P. 3.3.6

3.3.6 ID3 Vs CART

Table 3.3.1 : ID3 Vs CART

Sr. No.	Features	ID3	CART
1.	Acronym	Iterative Dichotomiser 3	Classification and Regression Tree
2.	Splitting Criteria	Information Gain	Gini Index
3.	Attribute Type	Handles only categorical data	Handles both categorical and numeric value
4.	Missing Values	Do not handle missing values	Handle missing values
5.	Pruning Strategy	No pruning is done	Post pruning is done
6.	Outlier Detection	Susceptible to outliers	Can handle outliers

3.4 NAIVE BAYESIAN CLASSIFICATION

- Bayesian classification is based on Bayes' Theorem.
- Bayesian classifiers are the statistical classifiers.
- Bayesian classifiers can predict class membership probabilities such as the probability that a given tuple belongs to a particular class.

3.4.1 Baye's Theorem

- Bayes's theorem is expressed mathematically by the following equation that is given below.

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

Where X and Y are the events and $P(Y) \neq 0$

- P(X|Y) is a **conditional probability** that describes the occurrence of event X given that Y is true.
- P(Y|X) is a **conditional probability** that describes the occurrence of event Y given that X is true.
- P(X) and P(Y) are the probabilities of observing X and Y independently of each other. This is known as the **marginal probability**.

3.4.2 Bayesian Interpretation

- In the Bayesian interpretation, probability determines a "degree of belief."
- Bayes theorem connects the degree of belief in a hypothesis before and after accounting for evidence.
- For example, lets us consider an example of the coin. If we toss a coin, then we get either head or tail, and the percent of occurrence of either head and tail is 50%. If the coin is flipped numbers of times, and the outcomes are observed, the degree of belief may rise, fall, or remain the same depending on the outcomes.
- For proposition X and evidence Y,
  - (a) P(X), the priori probability, is the primary degree of belief in X
  - (b) P(X|Y), the posterior probability, is the degree of belief having accounted for Y.
  - (c) The quotient $\frac{P(Y|X)}{P(Y)}$ represents the support Y provides for X.
- Bayes theorem can be derived from the conditional probability :

$$P(X|Y) = \frac{P(X \cap Y)}{P(Y)}, \text{ if } P(Y) \neq 0$$

$$P(Y|X) = \frac{P(Y \cap X)}{P(X)}, \text{ if } P(X) \neq 0$$

where $P(X \cap Y)$ is the joint probability of both X and Y being true, because

$$P(X \cap Y) = P(Y \cap X)$$

$$\text{Or, } P(X \cap Y) = P(X|Y)\,P(Y) = P(Y|X)\,P(X)$$

$$\text{Or, } P(X|Y) = \frac{P(Y|X)\,P(X)}{P(Y)}, \text{ if } P(Y) \neq 0$$

Ex. 3.4.1 : Apply statistical based algorithm to obtain the actual probabilities of each event to classify the new tuple as "Buys_Computer = Yes". Use the following data:

Classify X = (Age = "<=30", Income = "Medium", Student = "Yes", Credit_rating = "Fair").

Id	Age	Income	Student	Credit_rating	Buys_Computer
1	<=30	High	No	Fair	No
2	<=30	High	No	Excellent	No
3	31..40	High	No	Fair	Yes
4	>40	Medium	No	Fair	Yes
5	>40	Low	Yes	Fair	Yes
6	>40	Low	Yes	Excellent	No
7	31..40	Low	Yes	Excellent	Yes
8	<=30	Medium	No	Fair	No
9	<=30	Low	Yes	Fair	Yes
10	>40	Medium	Yes	Fair	Yes
11	<=30	Medium	Yes	Excellent	Yes
12	31..40	Medium	No	Excellent	Yes
13	31..40	High	Yes	Fair	Yes
14	>40	Medium	No	Excellent	No

Soln. :

Class label attribute is Buys_Computer.

C_1: Buys_Computer = Yes = 9 samples

C_2: Buys_Computer = No = 5 Samples

$\therefore P(C_1) = \frac{9}{14}$ and $P(C_2) = \frac{5}{14}$

Let event X_1 be Age = "<=30",

event X_2 be Income = "Medium",

event X_3 be Student = "Yes", and

event X_4 be Credit_rating = "Fair"

To compute : P(X|C1)

From Naïve Bayesian Classification

$$P(X|C_i) = \prod_{k=1}^{m} P(X_k|C_i)$$

$$P(X|C_1) = P(X_1|C_1)\,P(X_2|C_1)\,P(X_3|C_1)\,P(X_4|C_1)$$

$$P(X_1|C_1) = P\left(\frac{\text{Age <= 30}}{\text{Buys\_Computer = Yes}}\right) = \frac{2}{9}$$

$$P(X_2|C_1) = P\left(\frac{\text{Income = Medium}}{\text{Buys\_Computer = Yes}}\right) = \frac{4}{9}$$

$$P(X_3|C_1) = P\left(\frac{\text{Student = Yes}}{\text{Buys\_Computer = Yes}}\right) = \frac{6}{9}$$

$$P(X_4|C_1) = P\left(\frac{\text{Credit\_rating = Yes}}{\text{Buys\_Computer = Yes}}\right) = \frac{6}{9}$$

From Naïve Bayesian Classification

$$P(X|C_1) = \frac{2}{9}\times\frac{4}{9}\times\frac{6}{9}\times\frac{6}{9} = 0.044$$

$$P(X|C_1)\,P(C_1) = 0.044\times\frac{9}{14} = 0.028 \quad ...(A)$$

To compute: P(X|C2)

From Naive Bayesian Classification

$$P(X|C_i) = \prod_{k=1}^{m} P(X_k|C_i)$$

$$P(X|C_2) = P(X_1|C_2)\,P(X_2|C_2)\,P(X_3|C_2)\,P(X_4|C_2)$$

$$P(X_1|C_2) = P\left(\frac{\text{Age <= 30}}{\text{Buys\_Computer = No}}\right) = \frac{3}{5}$$

$$P(X_2|C_2) = P\left(\frac{\text{Income = Medium}}{\text{Buys\_Computer = No}}\right) = \frac{2}{5}$$

$$P(X_3|C_2) = P\left(\frac{\text{Student = Yes}}{\text{Buys\_Computer = No}}\right) = \frac{1}{5}$$

$$P(X_4|C_2) = P\left(\frac{\text{Credit\_rating = Yes}}{\text{Buys\_Computer = No}}\right) = \frac{2}{5}$$

From Naïve Bayesian Classification

$$P(X|C_2) = \frac{3}{5}\times\frac{2}{5}\times\frac{1}{5}\times\frac{2}{5} = 0.019$$

$$P(X|C_2)\,P(C_2) = 0.019\times\frac{5}{14} = 0.007 \quad ...(B)$$

Naive Bayesian classification will assign a sample X to class C_i if and only if

$P(C_i|X) > P(C_j|X)$

i.e. $\frac{P(X|C_i)P(C_i)}{P(X)} > \frac{P(X|C_j)P(C_j)}{P(X)}$

P(X) is constant for both classes C_i and C_j.

$\therefore P(X|C_i)\,P(C_i) > P(X|C_j)\,P(C_j)$ ...(C)

From (A), (B) and (C),

$P(X|C_1)\,P(C_1) > P(X|C_2)\,P(C_2)$

$\therefore$ We conclude that the unknown sample

X = (Age = "<=30", Income = "Medium", Student = "Yes", Credit_rating = "Fair") belongs to class

C_1 = Buys_Computer = Yes.

Ex. 3.4.2 : Apply statistical based algorithm to obtain the actual probabilities of each event to classify

X = (Dept = "Systems", Status = "Junior", Age = "26 -30").

Use the following table:

Dept	Status	Age	Salary	Count
Sales	Senior	31-35	46k – 50k	30
Sales	Junior	26-30	26k – 30k	40
Sales	Junior	26-30	31k – 35k	40
Systems	Junior	21-25	46k – 50k	20
Systems	Junior	31-35	66k – 70k	5
Systems	Junior	26-30	46k – 50k	5
Systems	Senior	41-45	66k – 70k	5

☑ **Soln. :**

Class label attribute is Salary.

C_1 = Salary between 46k – 50k = 55 Samples

C_2 = Salary between 26k – 30k = 40 Samples

C_3 = Salary between 31k – 40k = 40 Samples

C_4 = Salary between 66k – 70k = 10 Samples

$\therefore P(C_1) = \frac{55}{145}, P(C_2) = \frac{40}{145}, P(C_3) = \frac{40}{145}$

and $P(C_4) = \frac{40}{145}$

Let event X_1 be Dept = "Systems",

event X_2 be Status = "Junior", and

event X_3 be Age = "26 -30"

To compute: P(X|C1)

From Naïve Bayesian Classification

$$P(X|C_i) = \prod_{k=1}^{m} P(X_k|C_i)$$

$$P(X|C_1) = P(X_1|C_1)\,P(X_2|C_1)\,P(X_3|C_1)\,P(X_4|C_1)$$

$$P(X_1|C_1) = P\left(\frac{\text{Dept = Systems}}{\text{Salary 46 k – 50 k}}\right) = \frac{25}{55}$$

$$P(X_2|C_1) = P\left(\frac{\text{Status = Junior}}{\text{Salary 46 k – 50k}}\right) = \frac{25}{55}$$

$$P(X_3|C_1) = P\left(\frac{\text{Age = 26 – 30}}{\text{Salary 46 k – 50k}}\right) = \frac{5}{55}$$

From Naïve Bayesian Classification

$$P(X|C_1) = \frac{25}{55}\times\frac{25}{55}\times\frac{5}{55} = 0.019$$

$$P(X|C_1)P(C_1) = 0.019\times\frac{55}{145} = 0.007 \quad ...(A)$$

To compute: P(X|C2)

From Naïve Bayesian Classification

$$P(X|C_i) = \prod_{k=1}^{m} P(X_k|C_i)$$

$$P(X|C_2) = P(X_1|C_2)\,P(X_2|C_2)\,P(X_3|C_2)\,P(X_4|C_2)$$

$$P(X_1|C_2) = P\left(\frac{\text{Dept = Systems}}{\text{Salary 26k – 30 k}}\right) = \frac{0}{40}$$

$$P(X_2|C_2) = P\left(\frac{\text{Status = Junior}}{\text{Salary 26 k – 30k}}\right) = \frac{40}{40}$$

$$P(X_3|C_2) = P\left(\frac{\text{Age = 26 – 30}}{\text{Salary 26 k – 30k}}\right) = \frac{40}{40}$$

From Naïve Bayesian Classification

$$P(X|C_2) = \frac{0}{40}\times\frac{40}{40}\times\frac{40}{40} = 0$$

$$P(X|C_2)\,P(C_2) = 0\times\frac{40}{145} = 0 \quad ...(B)$$

To compute: P(X|C3)

From Naïve Bayesian Classification

$$P(X|C_i) = \prod_{k=1}^{m} P(X_k|C_i)$$

$$P(X|C_3) = P(X_1|C_3)\,P(X_2|C_3)\,P(X_3|C_3)\,P(X_4|C_3)$$

$$P(X_1|C_3) = P\left(\frac{\text{Dept = Systems}}{\text{Salary 31k – 35 k}}\right) = \frac{0}{40}$$

$$P(X_2|C_3) = P\left(\frac{\text{Status = Junior}}{\text{Salary 31 k – 35k}}\right) = \frac{40}{40}$$

$$P(X_3|C_3) = P\left(\frac{\text{Age = 26 – 30}}{\text{Salary 31 k – 35k}}\right) = \frac{40}{40}$$

From Naïve Bayesian Classification

$$P(X|C_3) = \frac{0}{40}\times\frac{40}{40}\times\frac{40}{40} = 0$$

$$P(X|C_3)\,P(C_3) = 0\times\frac{40}{145} = 0 \quad ...(C)$$

To compute: P(X|C4)

From Naïve Bayesian Classification

$$P(X|C_i) = \prod_{k=1}^{m} P(X_k|C_i)$$

$$P(X|C_4) = P(X_1|C_4)\,P(X_2|C_4)\,P(X_3|C_4)\,P(X_4|C_4)$$

$$P(X_1|C_4) = P\left(\frac{\text{Dept = Systems}}{\text{Salary 66k – 70 k}}\right) = \frac{10}{10}$$

$$P(X_2|C_4) = P\left(\frac{\text{Status = Junior}}{\text{Salary 66 k – 70k}}\right) = \frac{5}{10}$$

$$P(X_3|C_4) = P\left(\frac{\text{Age = 26 – 30}}{\text{Salary 66 k – 70k}}\right) = \frac{0}{10}$$

From Naïve Bayesian Classification

$$P(X|C_4) = \frac{10}{10}\times\frac{5}{10}\times\frac{0}{10} = 0$$

$$P(X|C_4)\,P(C_4) = 0\times\frac{10}{145} = 0 \quad ...(D)$$

Naïve Bayesian classification will assign a sample X to class C_i if and only if,

$P(C_i|X) > P(C_j|X)$

i.e. $\frac{P(X|C_i)\,P(C_i)}{P(X)} > \frac{P(X|C_j)\,P(C_j)}{P(X)}$

P(X) is constant for both classes C_i and C_j.

$\therefore P(X|C_i)\,P(C_i) > P(X|C_j)\,P(C_j)$ ...(E)

From (A), (B), (C), (D) and (E)

$P(X|C_1)\,P(C_1) > P(X|C_2)\,P(C_2) \geq P(X|C_3)\,P(C_3) \geq P(X|C_4)\,P(C_4)$

$\therefore$ We conclude that the unknown sample

X = (Dept = "Systems", Status = "Junior", Age = "26 -30") belongs to class **C_1 = Salary between 46k – 50k.**

Module 3

Ex. 3.4.3 : Given the training data for height classification, classify the tuple, t = <Rohit, M, 1.95> using Bayesian Classification.

Name	Gender	Height	Output
Kiran	F	1.6m	Short
Jatin	M	2m	Tall
Madhuri	F	1.09m	Medium
Manisha	F	1.88m	Medium
Shilpa	F	1.7m	Short
Bobby	M	1.85m	Medium
Kavita	F	1.6m	Short
Dinesh	M	1.7m	Short
Rahul	M	2.2m	Tall
Shree	M	2.1m	Tall
Divya	F	1.8m	Medium
Tushar	M	1.95m	Medium
Kim	F	1.9m	Medium
Aarti	F	1.8m	Medium
Rajashree	F	1.75m	Medium

Soln. :

From the above table, it is clear that there are 4 tuples classified as Short, 8 tuples classified as Medium and 3 tuples classified as Tall.

We divide the height attribute into six ranges as given below :

(0, 1.6], (1.6, 1.7], (1.7, 1.8], (1.8, 1.9], (1.9, 2.0] and (2.0, ∞]

Note : (a, b] indicates value greater than a and less than or equal to b.

Probabilities associated with attributes is given in table below.

Attribute	Value	Count			Probabilities		
		Short	Medium	Tall	Short	Medium	Tall
Gender	M	1	2	3	1/4	2/8	3/3
	F	3	6	0	3/4	6/8	0/3
	Total	4	8	3			
Height	(0, 1.6]	2	0	0	2/4	0	0
	(1.6, 1.7]	2	0	0	2/4	0	0
	(1.7, 1.8]	0	3	0	0	3/8	0
	(1.8, 1.9]	0	4	0	0	4/8	0
	(1.9, 2.0]	0	1	1	0	1/8	1/3
	(2.0, ∞]	0	0	2	0	0	2/3
	Total	4	8	3			

From the given training data, we estimate

P(Short) = 4/15

P(Medium) = 8/15

P(Tall) = 3/15

The unseen tuple is t = <Name = Rohit, Gender = M, Height = 1.95 >

$P(t|Short) \times P(Short) = P(M|Short) \times P(1.9.2.0|Short) \times P(Short)$

$= \frac{1}{4} \times 0 \times \frac{4}{15} = 0$

$P(t|Medium) \times P(Medium) = P(M|Medium) \times P((1.9.2.0)|Medium) \times P(Medium)$

$= \frac{2}{8} \times \frac{1}{8} \times \frac{8}{15} = 0.0167$

$P(t|tall) \times P(tall) = P(M|tall) \times P((1.9.2.0)|tall) \times P(tall)$

$= \frac{3}{3} \times \frac{1}{3} \times \frac{3}{15} = 0.067$

Based on the above probabilities, we classify the new tuple as **Tall** because it has the highest probability.

Ex. 3.4.4 : Given the training data for credit transaction. Classify a new transaction with (Income = Medium and credit = Good) using Naïve Bayes classification.

Transaction	Income	Credit	Decision
1	Very high	Excellent	AUTHORIZE
2	High	Good	AUTHORIZE
3	Medium	Excellent	AUTHORIZE
4	High	Good	AUTHORIZE
5	Very high	Good	AUTHORIZE
6	Medium	Excellent	AUTHORIZE
7	High	Bad	REQUEST ID
8	Medium	Bad	REQUEST ID
9	High	Bad	REJECT
10	Low	Bad	CALL POLICE

Soln. :

From the above table, it is clear that there are 6 tuples classified as AUTHORIZE, 2 tuples classified as REQUEST ID, 1 tuple classified as REJECT and 1 tuple classified as CALL POLICE.

From the given training data, we estimate

P(AUTHORIZE) = 6/10

P(REQUEST ID) = 2/10

P(REJECT) = 1/10

P(CALL POLICE) = 1/10

NOTES

Probabilities associated with attributes is given in table below.

Attribute	Value	Count				Probabilities			
		AUTHORIZE	REQUEST ID	REJECT	CALL POLICE	AUTHORIZE	REQUEST ID	REJECT	CALL POLICE
Income	Very High	2	0	0	0	2/6	0	0	0
	High	2	1	1	0	2/6	1/2	1	0
	Medium	2	1	0	0	2/6	1/2	0	0
	Low	0	0	0	1	0	0	0	1
	Total	6	2	1	1				
Credit	Excellent	3	0	0	0	3/6	0	0	0
	Good	3	0	0	0	3/6	0	0	0
	Bad	0	2	1	1	0	2/2	1	1
	Total	6	2	1	1				

The unknown tuple is t = <Income = Medium, Credit = Good>

$P(t|AUTHORIZE) \times P(AUTHORIZE) = P(Income = Medium|AUTHORIZE) \times P(Credit = Good|AUTHORIZE) \times P(AUTHORIZE)$

$= \frac{2}{6} \times \frac{3}{6} \times \frac{6}{10} = 0.1$

$P(t|REQ.ID) \times P(REQ.ID) = P(Income = Medium|REQ.ID) \times P(Credit = Good|REQ.ID) \times P(REQ.ID)$

$= \frac{1}{2} \times 0 \times \frac{2}{10} = 0$

$P(t|REJECT) \times P(REJECT) = P(Income = Medium| REJECT) \times P(Credit = Good|REJECT) \times P(REJECT)$

$= 0 \times 0 \times \frac{1}{10} = 0$

$P(t|CALL\ POLICE) \times P(CALL\ POLICE) = P(Income = Medium|CALL\ POLICE) \times P(Credit = Good|CALL\ POLICE) \times P(CALL\ POLICE)$

$= 0 \times 0 \times \frac{1}{10} = 0$

Based on the above probabilities, we classify the new tuple as **AUTHORIZE** because it has the highest probability.

UEx. 3.4.5 MU- May 2019, Dec. 2019

Using the given training dataset classify the following tuple using Naïve Bayes Algorithm.
<Homeowner: No, Marital Status: Married, Job experience:3>

Homeowner	Marital Status	Job experience (in years)	Defaulted
Yes	Single	3	No
No	Married	4	No
No	Single	5	No
Yes	Married	4	No
No	Divorced	2	Yes

Module 3

Homeowner	Marital Status	Job experience (in years)	Defaulted
No	Married	4	No
Yes	Divorced	2	No
No	Married	3	Yes
No	Married	3	No
Yes	Single	3	Yes

☑ **Soln. :**

From the above table, it is clear that there are 3 tuples classified as

Defaulted = Yes and 7 tuples classified as Defaulted = No.

From the given training data, we estimate

P(Defaulted = Yes) = 3/10

P(Defaulted = No) = 2/10

Probabilities associated with attributes is given in table below.

Attribute	Value	Count		Probabilities	
		Defaulted = Yes	Defaulted = No	Defaulted = Yes	Defaulted = No
Homeowner	Yes	1	3	1/3	3/7
	No	2	4	2/3	4/7
	Total	**3**	**7**		
Marital Status	Single	1	2	1/3	2/7
	Married	1	4	1/3	4/7
	Divorced	1	1	1/3	1/7
	Total	**3**	**7**		
Job experience (in years)	2	2	1	2/3	1/7
	3	1	2	1/3	2/7
	4	0	3	0	3/7
	5	0	1	0	1/7
	Total	**3**	**7**		

The unknown tuple is t = <Homeowner: No, Marital Status: Married, Job experience:3>

P(tlDefaulted = Yes) × P(Defaulted = Yes)

= P(Homeowner: Nol Defaulted = Yes) × P(Marital Status: Married| Defaulted

= Yes) × P(Job experience:3| Defaulted = Yes) × P(Defaulted = Yes)

$= \frac{1}{3} \times \frac{1}{3} \times \frac{1}{3} \times \frac{1}{3} = 0.011$

P(tlDefaulted = No) × P(Defaulted = No)

= P(Homeowner: Nol Defaulted = No) × P(Marital Status: Married| Defaulted = No) 4 P(Job experience:3| Defaulted

= No) × P(Defaulted = No)

$= \frac{4}{7} \times \frac{4}{7} \times \frac{2}{7} \times \frac{7}{10} = 0.032$

Based on the above probabilities, we classify the new tuple as **Defaulted = No** because it has the highest probability.

3.5 RULE BASED CLASSIFICATION : IF-THEN RULES

- Rule-based classifier makes use of a set of IF-THEN rules for classification.
- We can express a rule in the following form
- IF condition THEN conclusion
- Let us consider a rule R1,
- R1: IF age = youth AND student = yes THEN buy_computer = yes
- 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 consists of one or more attribute tests and these tests are logically ANDed.
- The consequent part consists of class prediction.
- To extract a rule from a decision tree –

1. One rule is created for each path from the root to the leaf node.
2. To form a rule antecedent, each splitting criterion is logically ANDed.
3. The leaf node holds the class prediction, forming the rule consequent.

3.6 REGRESSION

UQ. Explain Regression. Explain linear regression with example. **(MU - May 2019)**

- Regression is a data mining technique used to predict a range of numeric values (also called *continuous values*), given a particular dataset. For example, regression might be used to predict the cost of a product or service, given other variables.
- Data smoothing can also be done using regression.
- Regression can be classified as:

(a) Linear Regression :

It finds the best line to fit two variables or attributes so that one variable can be used to predict the other variable.

- Linear Regression : $y = \alpha + \beta x$

$$\beta = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}$$

where, $\bar{x}$ and $\bar{y}$ are the mean of variables x and y respectively.

- Linear regression is one of the easiest and most popular Supervised Machine Learning algorithms.
- It is a statistical method that is used for predictive analysis.
- Linear regression makes predictions for continuous/real or numeric variables such as sales, salary, age, product price, etc.
- Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (x) variables, hence called as linear regression.
- Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.
- The linear regression model provides a sloped straight line representing the relationship between the variables.

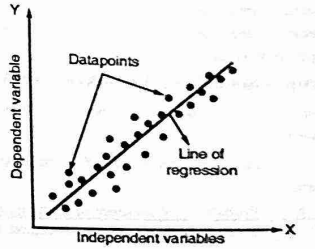


Fig. 3.6.1 : Linear Regression Model

(b) Multiple Linear Regression

- It is an extension of linear regression where more than two variables are involved and the data are fit to a multidimensional surface.

Multiple Linear Regression :

$y = \alpha + \beta_1 x_1 + \beta_1 x_2 + ... + \beta_n x_n$

- Here, y is the variable that is to be predicted, x is the variable used to predict y, α is the intercept and β is

Module 3

the slope.

$$\beta = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}$$

where, $\bar{x}$ and $\bar{y}$ are the mean of variables x and y respectively.

(a) Logistic Regression

- Logistic regression is basically a supervised classification algorithm.
- In a classification problem, the target variable (or output), y, can take only discrete values for given set of features (or inputs), X.
- **In logistic regression, the dependent variable is binary or dichotomous, i.e. it only contains data coded as 1 (TRUE, success, Infectious, etc.) or 0 (FALSE, failure, non-infectious, etc.).**
- The goal of logistic regression is to find the best fitting model to describe the relationship between the dichotomous characteristic of interest (dependent variable = response or outcome variable) and a set of independent (predictor or explanatory) variables.
- Logistic regression generates the coefficients (and its standard errors and significance levels) of a formula to predict a *logit transformation* of the probability of dependent variable:

  $\mathbf{logit(p) = b_0 + b_1X_1 + b_2X_2 + b_3X_3 + ... + b_kX_k}$

  where p is the probability of presence of the dependent variable.
- The logit transformation is defined as the logged odds, where

$$\text{Odds} = \frac{p}{1-p} = \frac{\text{Probability of presence of characteristics}}{\text{Probability of absence of characteristics}}$$

And $\text{log it(p)} = ln\left(\frac{p}{1-p}\right)$

- Rather than choosing parameters that minimize the sum of squared errors (like in ordinary regression), estimation in logistic regression chooses parameters that maximize the likelihood of observing the sample values.
- Logistic regression's ability to provide probabilities and classify new samples using continuous and discrete measurements makes it a popular machine learning method.
- **The logistic regression** technique involves dependent variable which can be represented in the binary (0 or 1, true or false, yes or no) values, means that the outcome could only be in either one form of two. For example, it can be utilized when we need to find the probability of successful or fail event.
- Here, the same formula as of linear regression is used with the additional sigmoid function, and the value of Y ranges from 0 to 1.
- Logistic regression equation :

Linear regression $Y = b_0 + b_1 \times X_1 + b_2 \times X_2 + ... + b_k \times X_k$

Sigmoid Function $P = \frac{1}{1 + e^{-Y}}$

By putting Y in Sigmoid function, we get the following result.

$$ln\left(\frac{p}{1-p}\right) = b_0 + b_1 \times X_1 + b_2 \times X_2 + ... + b_k \times X_k$$

- The logistic function has asymptotes at 0 and 1, and it crosses the y-axis at 0.5 as shown in Fig. 3.6.2.

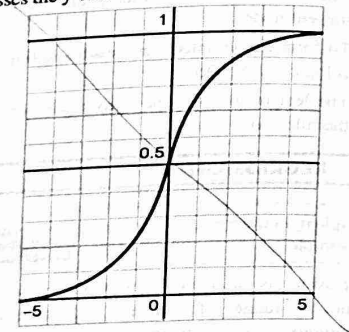


Fig. 3.6.2 : Logistic Function

- A prediction function in logistic regression returns the probability of our observation being positive, True, or "Yes". We call this class 1 and its notation is *P(class=1)*.
- In order to map this to a discrete class (true/false, cat/dog), we select a threshold value or tipping point above which we will classify values into class 1 and below which we classify values into class 2.

$P \geq 0.5$, class = 1

$P < 0.5$, class = 0

Ex. 3.6.1 : Develop a model to predict the salary of college graduates with 10 years of work experience using linear regression.

Years of Experience (x)	3	8	9	13	3	6	11	21	1	16
Salary in $100 (y)	30	57	64	72	36	43	59	90	20	83

☑ **Soln. :**

Linear Regression : $y = \alpha + \beta x$

where

$$\beta = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}$$

$$\bar{x} = \frac{1}{10}\sum_{i=1}^{10} x_i = 9.1 \quad \bar{y} = \frac{1}{10}\sum_{i=1}^{10} y_i = 55.4$$

x	y	$x_i - \bar{x}$	$y_i - \bar{y}$	$(x_i - \bar{x})(y_i - \bar{y})$	$(x_i - \bar{x})^2$
3	30	- 6.1	- 25.4	154.94	37.21
8	57	- 1.1	1.6	- 1.76	1.21
9	64	- 0.1	8.6	- 0.86	0.01
13	72	3.9	16.6	64.74	15.21
3	36	- 6.1	- 19.4	118.34	37.21
6	43	- 3.1	- 12.4	38.44	9.61
11	59	1.9	3.6	6.84	3.61
21	90	11.9	34.6	411.74	141.61
1	20	- 8.1	- 35.4	286.74	65.61
16	83	6.9	27.6	190.44	47.61
				$\Sigma = 1269.6$	$\Sigma = 358.9$

$$\beta = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2} = 3.54$$

$$\alpha = \bar{y} - \beta \bar{x} = 55.4 - 3.54 \times 9.1 = 23.19$$

For x =10 years, $y = \alpha + \beta x = 23.19 + 3.54 \times 10 = 58.586$ (in $ 100)

Ex. 3.6.2 : The following table shows the midterm and final exam grades obtained for students in Database course. Use the method of least squares using regression to predict the final exam grade of a student who received 86 on the midterm exam.

Midterm Exam (x)	72	50	81	74	94	86	59	83	65	33	88	81
Final Exam (y)	84	63	77	78	90	75	49	79	77	52	74	90

☑ **Soln. :**

Linear Regression : $y = \alpha + \beta x$

where

$$\beta = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2}$$

Module 3

$\bar{x} = \sum_{i=1}^{12} x_i = 72.17$; $\bar{y} = \sum_{i=1}^{12} y_i = 74$

Midterm Exam (x)	Final Exam (y)	$x_i - \bar{x}$	$y_i - \bar{y}$	$(x_i - \bar{x})^2$	$(x_i - \bar{x})(y_i - \bar{y})$
72	84	– 0.17	10	0.03	– 1.67
50	63	– 22.17	– 11	491.36	243.83
81	77	8.83	3	78.03	26.50
74	78	1.83	4	3.36	7.33
94	90	21.83	16	476.69	349.33
86	75	13.83	1	191.36	13.83
59	49	– 13.17	– 25	173.36	329.17
83	79	10.83	5	117.36	54.17
65	77	– 7.17	3	51.36	– 21.50
33	52	– 39.17	– 22	1534.03	861.67
88	74	15.83	0	250.69	0.00
81	90	8.83	16	78.03	141.33
				$\Sigma = 3445.67$	$\Sigma = 2004$

$$\beta = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2} = 0.58$$

$$\alpha = \bar{y} - \beta \bar{x} = 74 - 0.58 \times 72.17 = 32.14$$

∴ The linear regression line becomes : $y = 32.14 + 0.58 \times x$

The final exam grade of a student who received 86 on the midterm exam :

$y = 32.14 + 0.58 \times 86 = 82.02 \approx 82$ marks

Ex. 3.6.3 : Say we're given data on student exam results and our goal is to predict whether a student will pass or fail based on number of hours slept and hours spent studying. We have two features (hours slept, hours studied) and two classes: passed (1) and failed (0).

Studied (x_1)	Slept (x_2)	Passed (y)
4.85	9.63	0
8.62	3.23	1
5.43	8.23	0
9.21	6.34	1

Let $b_0 = -0.406$, $b_1 = 0.8525$, $b_2 = -1.105$. Determine the predicted value of y using Logistic Regression.

☑ **Soln. :**

Logistic regression equation

Linear regression $y = b_0 + b_1 \times x_1 + b_2 \times x_2 + ... + b_k \times x_k$

Sigmoid Function : $P = \frac{1}{1 + e^{-Y}}$

For 1st input tuple, $x_1 = 4.85$ and $x_2 = 9.63$,

$$y = \frac{1}{1 + e^{-(b_0 + b_1 x_1 + b_2 x_2)}} = \frac{1}{1 + e^{-(-0.406 + 0.8525 * 4.85 - 1.105 * 9.63)}} = 0.0022$$

For 2nd input tuple, $x_1 = 8.62$ and $x_2 = 3.23$,

$$y = \frac{1}{1 + e^{-(b_0 + b_1 x_1 + b_2 x_2)}} = \frac{1}{1 + e^{-(-0.406 + 0.8525 * 8.62 - 1.105 * 3.23)}} = 0.96$$

For 3rd input tuple, $x_1 = 5.43$ and $x_2 = 8.23$,

$$y = \frac{1}{1 + e^{-(b_0 + b_1 x_1 + b_2 x_2)}} = \frac{1}{1 + e^{-(-0.406 + 0.8525 * 5.43 - 1.105 * 8.23)}} = 0.0076$$

For 4th input tuple, $x_1 = 9.21$ and $x_2 = 6.34$,

$$y = \frac{1}{1 + e^{-(b_0 + b_1 x_1 + b_2 x_2)}} = \frac{1}{1 + e^{-(-0.406 + 0.8525 * 4.85 - 1.105 * 9.63)}} = 0.6082$$

In Logistic Regression,

$P \geq 0.5$, class = 1

$P < 0.5$, class = 0

The predicted and actual value are shown in table below:

Studied (x_1)	Slept (x_2)	Passed (y)	Fitted Value	Prediction
4.85	9.63	0	0.0022	0
8.62	3.23	1	0.96	1
5.43	8.23	0	0.0076	0
9.21	6.34	1	0.6082	1

Module 3

3.6.1 Linear Regression Vs Logistic Regression

Sr. No.	Linear Regression	Logistic Regression
1	Linear regression is used to predict the continuous dependent variable using a given set of independent variables.	Logistic Regression is used to predict the categorical dependent variable using a given set of independent variables.
2	Linear Regression is used for solving Regression problem.	Logistic regression is used for solving Classification problems.
3	In Linear regression, we predict the value of continuous variables.	In logistic Regression, we predict the values of categorical variables.
4	In linear regression, we find the best fit line, by which we can easily predict the output.	In Logistic Regression, we find the S-curve by which we can classify the samples.
5	Least square estimation method is used for estimation of accuracy.	Maximum likelihood estimation method is used for estimation of accuracy.
6	The output for Linear Regression must be a continuous value, such as price, age, etc.	The output of Logistic Regression must be a Categorical value such as 0 or 1, Yes or No, etc.
7	In Linear regression, it is required that relationship between dependent variable and independent variable must be linear.	In Logistic regression, it is not required to have the linear relationship between the dependent and independent variable.
8	In linear regression, there may be collinearity between the independent variables.	In logistic regression, there should not be collinearity between the independent variable

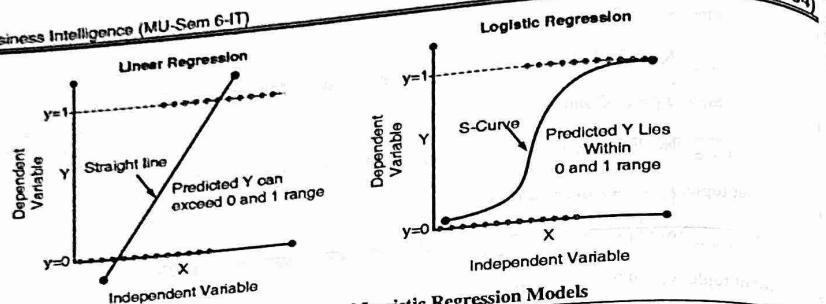


Fig. 3.6.3 : Linear Regression and Logistic Regression Models

3.7 ACCURACY AND ERROR MEASURES

- In data mining, classification involves the problem of predicting which category or class a new observation belongs in.
- The derived model (classifier) is based on the analysis of a set of training data where each data is given a class label.
- The trained model (classifier) is then used to predict the class label for new, unseen data.
- To understand classification metrics, one of the most important concepts is the confusion matrix.
- The different classifier evaluation measures are discussed below:

1. **Confusion Matrix** : It is a useful tool for analyzing how well your classifier can recognize tuples of different classes. It is also called as contingency matrix. Each row in a confusion matrix represents an actual class, while each column represents a predicted class.

The 2 × 2 confusion matrix is denoted as:

		Predicted Class	
		1	0
Actual Class	1	TP	FN
	0	FP	TN

- **TP :** It represents the values which are predicted to be true and are actually true.
- **TN :** It represents the values which are predicted to be false and are actually false.
- **FP :** It represents the values which are predicted to be true, but are false. Also called Type I error.
- **FN :** It represents the values which are predicted to be false, but are true. Also called Type II error.

2. **Sensitivity :** Also called the true positive recognition rate. It is proportion of positive tuples that are correctly identified.

$$\text{Sensitivity} = \frac{TP}{TP + FN}$$

3. **Specificity :** Also called the true negative rate. It is proportion of negative tuples that are correctly identified.

$$\text{Specificity} = \frac{TN}{TN + FP}$$

4. **Accuracy :** The accuracy of a classifier on a given test set is the percentage of test set tuples that are correctly classified by the classifier. It is also referred to as the overall recognition rate of the classifier.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

5. **Precision :** It is the measure of exactness. It determines what percentage of tuples labelled as positive are actually positive.

$$\text{Precision} = \frac{TP}{TP + FP}$$

6. **Recall :** It is the measure of completeness. It determines what percentage of positive tuples are labelled as positive.

$$\text{Recall} = \frac{TP}{TP + FN}$$

7. **F-Score :** It is the harmonic mean of precision and recall. It gives equal weight to precision and recall. It is also called F-measure or F_1 score.

$$F = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

8. **F_β Score:** It is the weighted measure of precision and recall. It assigns β times as much weight to recall as to precision. Commonly used F_β measures are F_2 (which weights recall twice as much as precision) and $F_{0.5}$ (which weights precision twice as much as recall).

$$F_\beta = \frac{(1 + \beta^2) \times \text{Precision} \times \text{Recall}}{\beta^2 \times \text{Precision} + \text{Recall}}$$

9. **Error Rate :** It is also called misclassification rate of a classifier and is simply (1 – Accuracy).

UEx. 3.7.1 (MU - May 2019)
Calculate Accuracy, Precision and Recall for the following Confusion Matrix.

Cancer Classes	Yes	No	Total
Yes	90	210	300
No	140	9560	9700
Total	230	9770	10000

Soln. :

The 2 × 2 Confusion Matrix is denoted as:

		Predicted Class	
		Yes	No
Actual Class	Yes	TP	FN
	No	FP	TN

Given Confusion Matrix is

Cancer Classes	Yes	No	Total
Yes	90	210	300
No	140	9560	9700
Total	230	9770	10000

Comparing the above matrices, we get

TP = 90, FN = 210, FP = 140 and TN = 9560

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{90 + 9560}{90 + 9560 + 140 + 210} = 0.965 = 96.5\%$$

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{90}{90 + 140} = 0.3913 = 39.13\ \%$$

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{90}{90 + 210} = 0.3 = 30\%$$

3.8 EVALUATING THE ACCURACY OF A CLASSIFIER

Besides the evaluation measure discussed above, other techniques to evaluate the accuracy of a classifier are discussed below.

3.8.1 Holdout

- In this method, the mostly large dataset is randomly divided to three subsets.

1. **Training set** is a subset of the dataset used to build predictive models.
2. **Validation set** is a subset of the dataset used to assess the performance of model built in the training phase. It provides a test platform for fine tuning model's parameters and selecting the best-performing model. Not all modeling algorithms need a validation set.
3. **Test set** or unseen examples is a subset of the dataset to assess the likely future performance of a model. If a model fit to the training set much better than it fits the test set, overfitting is probably the cause.

- Typically, two-thirds of the data are allocated to the training set and the remaining one-third is allocated to the test set.

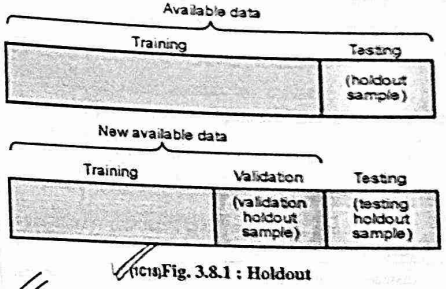


Fig. 3.8.1 : Holdout

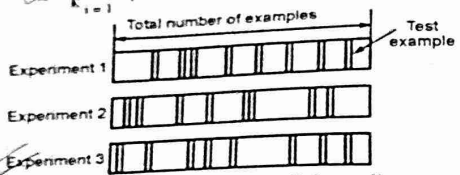
3.8.2 Random Subsampling

- It is a variation of the holdout method. The holdout method is repeated k times.
- It involves randomly splitting the data into a training and a test set.

Module 3

- The data is trained on the training set and the mean square error (MSE) is obtained from the predictions on the test set.
- This method is not recommended because the MSE would depend on the split. So a new split can give you a new MSE and then you don't know which to trust.
- The overall accuracy is calculated by taking the average of the accuracies obtained from each iteration.

$$E = \frac{1}{k}\sum_{i=1}^{k} E_i$$



(IC15)**Fig. 3.8.2 : Random Subsampling**

3.8.3 Cross Validation

- When only a limited amount of data is available, to achieve an unbiased estimate of the model performance we use k-fold cross-validation.
- In k-fold cross-validation, we divide the data into k subsets of equal size.
- We build models k times, each time leaving out one of the subsets from training and use it as the test set.
- If k equals the sample size, this is called "leave-one-out".

3.8.4 Bootstrapping

- Bootstrapping is a technique used to make estimations from data by taking an average of the estimates from smaller data samples.
- The bootstrap method involves iteratively resampling a dataset with replacement.
- Instead of only estimating our statistic once on the complete data, we can do it many times on a re-sampling (with replacement) of the original sample.
- Repeating this re-sampling multiple times allows us to obtain a vector of estimates.
- We can then compute variance, expected value, empirical distribution, and other relevant statistics of these estimates.

3.9 INTRODUCTION TO ENSEMBLE METHODS

- Ensemble methods are techniques that aim at improving the accuracy of results in models by combining multiple models instead of using a single model.
- The combined models increase the accuracy of the results significantly. This has boosted the popularity of ensemble methods in machine learning.
- Ensemble methods are ideal for regression and classification, where they reduce bias and variance to boost the accuracy of models.

Types of Ensemble Methods

1. Bagging

- Bagging, the short form for bootstrap aggregating, is mainly applied in classification and regression.
- Bagging is based on Bootstrap sampling technique.
- In each iteration a different set of bootstrap sample is generated for constructing the individual classifier of the same algorithm.
- Bootstrap sampling technique selects the data item randomly with replacement, i.e. some instance, can be repeated or some of them can be missed out from the original dataset during the sampling step.
- The next step of bagging is to combine all the classifiers constructed in previous phase.
- Bagging combines the result of the classifiers with the help of voting to make final prediction.
- Bagging is an effective ensemble technique for unstable learning algorithms where small changes in the training data set results in big changes in predictions for e.g. Decision Tree, Neural Network, etc.

2. Boosting

- Boosting boosts the performance of the weak classifier to a strong level.
- It generates sequential learning classifiers using resampling (reweighting) the data instances.
- Initially equal uniform weights are assigned to all the instances.

- During each learning phase a new hypothesis is learned and the instances are reweighted such that system concentrates on correctly classified instances having lower weight and instances that have not been correctly classified during this phase having higher weights.
- It selects the wrongly classified instance, so that they can be classified correctly during the next learning step. This process continuous till the construction of the last classifier.
- Finally, the results of all the classifiers are combined using majority voting to find the final prediction.
- AdaBoost is a more general version of the Boosting algorithm.

3. Random Forest

- Random Forest is proposed particularly for trees.
- It is the combination of bagging and random subspace method for inducing the tree.
- It is similar to bagging except that each model is a random tree rather than a single model and each tree is grown according to the bootstrap sample of the training set to N.
- Another random step is used to split each node.
- A small subset of feature m is selected randomly ((m < M) rather than considers all possible splits M, and the best split is chosen from this subset.
- Final classification is done by majority vote across trees.

4. AdaBoost Algorithm

- AdaBoost algorithm, short for Adaptive Boosting, is a Boosting technique that is used as an Ensemble Method in Machine Learning.
- It is called Adaptive Boosting as the weights are re-assigned to each instance, with higher weights to incorrectly classified instances.
- Boosting is used to reduce bias as well as the variance for supervised learning. It works on the principle where learners are grown sequentially.
- Except for the first, each subsequent learner is grown from previously grown learners. In simple words, weak learners are converted into strong ones.
- AdaBoost algorithm also works on the same principle as boosting, but there is a slight difference in working.
- It makes n number of decision trees during the training period of data.
- As the first decision tree/model is made, the record which is incorrectly classified during the first model is given more priority. Only these records are sent as input for the second model.
- The process will go on until we specify a number of base learners we want to create.
- When the random forest is used, the algorithm makes n number of trees. It makes proper trees that consist of a start node with several leaves nodes. Some trees might be bigger than others, but there is no fixed depth in a random forest.
- In AdaBoost, the algorithm only makes a node with two leaves, and this is known as Stump.
- These stumps are weak learners, and boosting techniques prefer this. The order of stumps is very important in AdaBoost. The error of the first stump influences how the other stump is made.
- So, in AdaBoost algorithm also, the majority of votes take place between the stumps, the same as in random trees.
- Final classification is done by majority vote across trees.

Module 3

3.10 NEAREST NEIGHBOR CLASSIFIER

- K-Nearest Neighbor is one of the simplest algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data.