USN | | | | | | | | | | |

Internal Assessment Test 2 – November 2020

Sub:	Machine Learning					Sub Code:	17CS73	Branch:	ISE	
Date:	05/11/2020	Duration:	90 min's	Max Marks:	50	Sem / Sec:	7 / A,B			OBE

	Answer the below MCQs (10 Marks)	MARKS	CO	RBT
1	Pre-pruning the decision tree may result in ________________ a. Over fitting **b. Under fitting** c. Perfect fitting d. None of these	1	CO2	L1
2	Below are the 8 actual values of target variable in the training file. [0, 0, 0, 1, 1, 1,1,1] What is the entropy of the target variable? **a. -(5/8 log(5/8) + 3/8 log(3/8))** b. 5/8 log(5/8) + 3/8 log(3/8) c. 3/8 log(5/8) + 5/8 log(3/8) d. 5/8 log(3/8) – 3/8 log(5/8)	1	CO2	L2
3	Entropy value of _________________ represents that data sample is pure or homogeneous(All positive samples or all negative samples) a) -1 **b) 0** c) 0.5 d) 1	1	CO2	L1
4	Which of the following is preferred strategy for building the most optimal decision tree a.Pre-pruning **b.Post pruning**	1	CO2	L1
5	A neuron with 3 inputs has the weight vector[0.2, -0.1, 0.1] and bias ɵ =0.If the input vector is x =[0.2, 0.4, 0.2] then the total input to the neuron is : 0.2 1.0 **0.02** -1.0	1	CO3	L2
6	Artificial neural network used for Pattern Recognition Classification Clustering **All of these**	1	CO3	L1
7	Internal state of neuron is called ___________, is the function of the inputs the neurons receives Weight **Activation or activity level of neuron** Bias None of these	1	CO3	L1
8	A 4-input neuron has weights 1, 2, 3 and 4. The transfer function is linear with the constant of proportionality being equal to 2. The inputs are 4, 10, 5 and 20 respectively. The output will be: a) **238** b) 76 c) 119 d) 123	1	CO3	L2
	Which of the below are not considered as the remarks on the back propagation algorithm	1	CO3	L2

<table>
<tr><td>9</td><td>Convergence and local minima
Hypothesis space search and inductive bias
Hidden layer representation
Output layer representation</td><td></td><td></td><td></td></tr>
<tr><td>10</td><td>What set of functions can be represented by feed-forward networks?
Boolean functions
Continuous functions
Arbitrary functions
Activation functions</td><td>1</td><td>CO3</td><td>L2</td></tr>
<tr><td colspan="5">**Answer any 4 of the following questions (40 Marks)**</td></tr>
<tr><td>11 (a)</td><td>Construct Decision trees to represent the Boolean Functions:
a) A && ~ B
b) A V [B && C]
c) A XOR B
d) [A&&B] V [C&&D]

Scheme:
• Construction of decision tree for each Boolean expression 2M = 2*4 =8M</td><td>[08]</td><td>CO2</td><td>L3</td></tr>
<tr><td>(b)</td><td>Discuss the two approaches to prevent over fitting the data.
Scheme:
Explanation of pre-pruning and post pruning 2M = 2*1 =2M</td><td>[02]</td><td>CO2</td><td>L1</td></tr>
<tr><td>12</td><td>Create and explain the decision tree for the following transactions using ID3 algorithm.
<table>
<tr><th>Day</th><th>A1</th><th>A2</th><th>A3</th><th>Classification</th></tr>
<tr><td>1</td><td>True</td><td>Hot</td><td>High</td><td>No</td></tr>
<tr><td>2</td><td>True</td><td>Hot</td><td>High</td><td>No</td></tr>
<tr><td>3</td><td>False</td><td>Hot</td><td>High</td><td>Yes</td></tr>
<tr><td>4</td><td>False</td><td>Cool</td><td>Normal</td><td>Yes</td></tr>
<tr><td>5</td><td>False</td><td>Cool</td><td>Normal</td><td>Yes</td></tr>
<tr><td>6</td><td>True</td><td>Cool</td><td>High</td><td>No</td></tr>
<tr><td>7</td><td>True</td><td>Hot</td><td>High</td><td>No</td></tr>
<tr><td>8</td><td>True</td><td>Hot</td><td>Normal</td><td>Yes</td></tr>
<tr><td>9</td><td>False</td><td>Cool</td><td>Normal</td><td>Yes</td></tr>
<tr><td>10</td><td>False</td><td>Cool</td><td>High</td><td>No</td></tr>
</table>
Scheme:
• Calculating the overall entropy 2M
• Calculating individual attribute gain values along with entropy 4M
• Identifying root node and sub nodes 2M
Constructing final decision tree 2M</td><td>[10]</td><td>CO2</td><td>L3</td></tr>
<tr><td>13</td><td>Write an algorithm for back propagation which uses stochastic gradient descent method. Also derive the back propagation rule considering the output layer and training rule for output unit weights.
Scheme:
• Back propagation algorithm 4M
• Deriving the derivatives rule 6M</td><td>[10]</td><td>CO3</td><td>L2</td></tr>
<tr><td>14(a)</td><td>Draw the perceptron network with the notation. Explain the following components of artificial neural networks
i) Perceptrons ii) Representational power of Perceptrons iii) Perceptron training rule
Scheme:
• Solving perceptron A and B
• Justification</td><td>[07]</td><td>CO3</td><td>L2</td></tr>
<tr><td>b</td><td>Consider two perceptrons defined by the threshold expression w0+w1x1+w2x2 >0.
Perceptron A has weight values w0=1, w1=2, w2=1.</td><td>[03]</td><td>CO3</td><td>L4</td></tr>
</table>

Perceptron B has weight values w0=0, w1=2, w2=1.
True or False? Perceptron A has more general than perceptron B. Justify your answer

Scheme:

- Solving perceptron A and B 2M
- Justification 1M

15 | [10] | CO3 | L3

Create and explain the decision tree for the following transactions using ID3 algorithm.

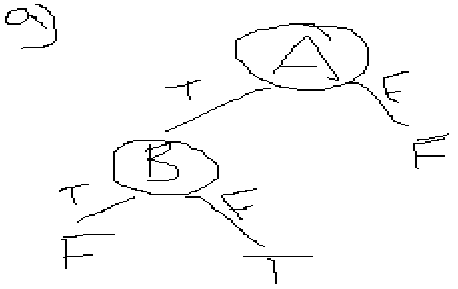
Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

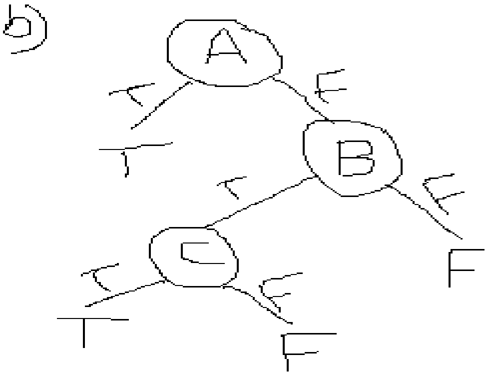
Scheme:

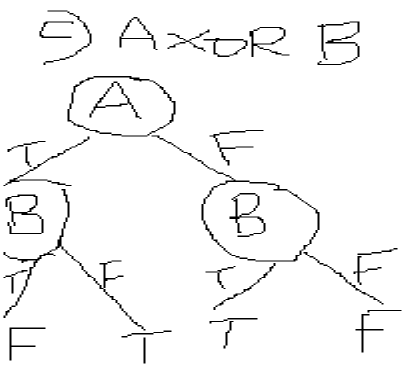
- Calculating the overall entropy 2M
- Calculating individual attribute gain values along with entropy 4M
- Identifying root node and sub nodes 2M
- Constructing final decision tree 2M

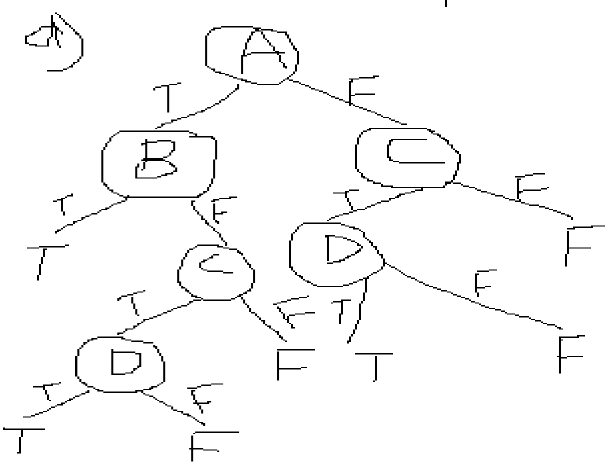
Solutions:

11)a) a) A && ~ B b) A V [B && C]
c) A XOR B d) [A&&B] V [C&&D]









11b)

Approaches to avoiding overfitting in decision tree learning

- **Pre-pruning (avoidance):** Stop growing the tree earlier, before it reaches the point where it perfectly classifies the training data
- **Post-pruning (recovery):** Allow the tree to overfit the data, and then post-prune the tree

1.1. Reduced-error pruning :

- ***Pruning*** a decision node consists of removing the subtree rooted at that node, making it a leaf node, and assigning it the most common classification of the training examples affiliated with that node
- Nodes are removed only if the resulting pruned tree performs no worse than-the original over the validation set.
- Reduced error pruning has the effect that any leaf node added due to coincidental regularities in the training set is likely to be pruned because these same coincidences are unlikely to occur in the validation set

1.2. Rule Post-Pruning :

Rule post-pruning is successful method for finding high accuracy hypotheses

Rule post-pruning involves the following steps:

1. Infer the decision tree from the training set, growing the tree until the training data is fit as well as possible and allowing overfitting to occur.
2. Convert the learned tree into an equivalent set of rules by creating one rule for each path from the root node to a leaf node.
3. Prune (generalize) each rule by removing any preconditions that result in improving its estimated accuracy.
4. Sort the pruned rules by their estimated accuracy, and consider them in this sequence when classifying subsequent instances.

For example, consider the decision tree above. The leftmost path of the tree in below figure is translated into the rule.

IF (Outlook = Sunny) ^ (Humidity = High) THEN *PlayTennis* = No

Given the above rule, rule post-pruning would consider removing the preconditions (Outlook = Sunny) and (Humidity = High)

It would select whichever of these pruning steps produced the greatest improvement in estimated rule accuracy

---

12)

Entropy(S) = 1 bcz all the samples are equal

Gain(S, A1) = Entropy(S) – {(5/10)*Entropy (Strue) + (5/10)* Entropy (Sfalse)}

Entropy(Strue) = -(1/5)log2(1/5) – (4/5)log2(4/5) = 0.72

Entropy(Sfalse) = -(4/5)log2(4/5)-(1/5)log2(1/5) = 0.72

Gain(S,A1) = 1 – {(5/10)*0.72 + (5/10)*0.72} = 1.129

Gain(S,A2) = Entropy(S) –{(5/10)*Entropy(Shot) +(5/10)*Entropy(SCool)}

Entropy(Shot)= -(2/5)log(2/5) – (3/5)log(3/5) = 0.966

Entropy(Scool) = -(3/5)log(3/5)-(2/5)log(2/5) = 0.966

Gain(S,A2) = 1- {(5/10)*0.966 + (5/10)*0.966} = 1.233

Gain(S,A3) = Entropy(S) –{(6/10)*Entropy(Shigh) +(4/10*Entropy(Snormal)}

Entropy(Shigh) = -(1/6)log(1/6) –(5/6)log(5/6) = 0.647

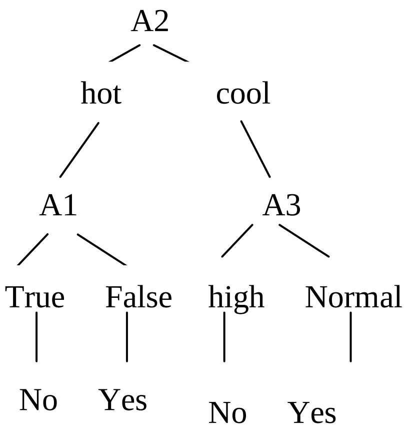
Entropy(Snormal) = 0 bca all are belonging to same class

Gain(S,A3) = 1 – ((6/10)* 0.647 + (4/10)*0} = 0.6118

Gain(S,A1) = 1.129

Gain(S,A2) =1.233

Gain(S,A3) = 0.6118 , so A2 will be chosen as root node



---

13)

BACKPROPAGATION (*training_example*, η, n_{in}, n_{out}, n_{hidden})

Each training example is a pair of the form $(\vec{x}, \vec{t})$, *where* $(\vec{x})$ *is the vector of network input values,* $(\vec{t})$ *and is the vector of target network output values.*
η *is the learning rate (e.g., .05).* n_i, *is the number of network inputs,* n_{hidden} *the number of units in the hidden layer, and* n_{out} *the number of output units.*
The input from unit i into unit j is denoted x_{ji}, *and the weight from unit i to unit j is denoted* w_{ji}

- Create a feed-forward network with n_i inputs, n_{hidden} hidden units, and n_{out} output units.
- Initialize all network weights to small random numbers
- Until the termination condition is met, Do
  - For each $(\vec{x}, \vec{t})$, in training examples, Do

    Propagate the input forward through the network:
    1. Input the instance $\vec{x}$, to the network and compute the output o_u of every unit u in the network.

Propagate the errors backward through the network:

2. For each network output unit k, calculate its error term δ_k

$$\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k)$$

3. For each hidden unit h, calculate its error term δ_h

$$\delta_h \leftarrow o_h(1 - o_h) \sum_{k \in outputs} w_{h,k}\delta_k$$

4. Update each network weight w_{ji}

$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$$

Where

$$\Delta w_{ji} = \eta \delta_j x_{i,j}$$

Derivation of the BACKPROPAGATION Rule

- Deriving the stochastic gradient descent rule: Stochastic gradient descent involves iterating through the training examples one at a time, for each training example d descending the gradient of the error E_d with respect to this single example

$$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial w_{ji}} \quad \text{.........equ. (1)}$$

where, E_d is the error on training example d, summed over all output units in the network

$$E_d(\vec{w}) \equiv \frac{1}{2} \sum_{k \in output} (t_k - o_k)^2$$

- For each training example d every weight w_{ji} is updated by adding to it Δw_{ji}

Here outputs is the set of output units in the network, t_k is the target value of unit ***k*** for training example ***d***, and o_k is the output of unit ***k*** given training example ***d***.

The derivation of the stochastic gradient descent rule is conceptually straightforward, but requires keeping track of a number of subscripts and variables

- x_{ji} = the i^{th} input to unit j
- w_{ji} = the weight associated with the i^{th} input to unit j
- $net_j = \Sigma_i w_{ji} x_{ji}$ (the weighted sum of inputs for unit j)
- o_j = the output computed by unit j
- t_j = the target output for unit j
- σ = the sigmoid function
- outputs = the set of units in the final layer of the network
- Downstream(j) = the set of units whose immediate inputs include the output of unit j

derive an expression for $\frac{\partial E_d}{\partial w_{ji}}$ in order to implement the stochastic gradient descent rule

seen in Equation $\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial w_{ji}}$

notice that weight w_{ji} can influence the rest of the network only through net_j.

Use chain rule to write

$$\frac{\partial E_d}{\partial w_{ji}} = \frac{\partial E_d}{\partial net_j} \frac{\partial net_j}{\partial w_{ji}}$$

$$= \frac{\partial E_d}{\partial net_j} x_{ji} \quad \text{......equ(2)}$$

Derive a convenient expression for $\frac{\partial E_d}{\partial net_j}$

Consider two cases: The case where unit j is an output unit for the network, and the case where j is an internal unit (hidden unit).

Case 1: Training Rule for Output Unit Weights.

w_{ji} can influence the rest of the network only through net_j , net_j can influence the network only through o_j. Therefore, we can invoke the chain rule again to write

$$\frac{\partial E_d}{\partial net_j} = \frac{\partial E_d}{\partial o_j}\frac{\partial o_j}{\partial net_j} \qquad \text{.....equ(3)}$$

To begin, consider just the first term in Equation (3)

$$\frac{\partial E_d}{\partial o_j} = \frac{\partial}{\partial o_j}\frac{1}{2}\sum_{k\in outputs}(t_k - o_k)^2$$

The derivatives $\frac{\partial}{\partial o_j}(t_k - o_k)^2$ will be zero for all output units k except when $k = j$. We therefore drop the summation over output units and simply set $k = j$.

$$\begin{aligned}\frac{\partial E_d}{\partial o_j} &= \frac{\partial}{\partial o_j}\frac{1}{2}(t_j - o_j)^2 \\ &= \frac{1}{2}2(t_j - o_j)\frac{\partial(t_j - o_j)}{\partial o_j} \\ &= -(t_j - o_j) \qquad \text{......equ(4)}\end{aligned}$$

Next consider the second term in Equation (3). Since $o_j = \sigma(net_j)$, the derivative $\frac{\partial o_j}{\partial net_j}$ is just the derivative of the sigmoid function, which we have already noted is equal to $\sigma(net_j)(1 - \sigma(net_j))$. Therefore,

$$\frac{\partial o_j}{\partial net_j} = \frac{\partial \sigma(net_j)}{\partial net_j} = o_j(1 - o_j) \quad \text{.......equ(5)}$$

Substituting expressions (4) and (5) into (3), we obtain

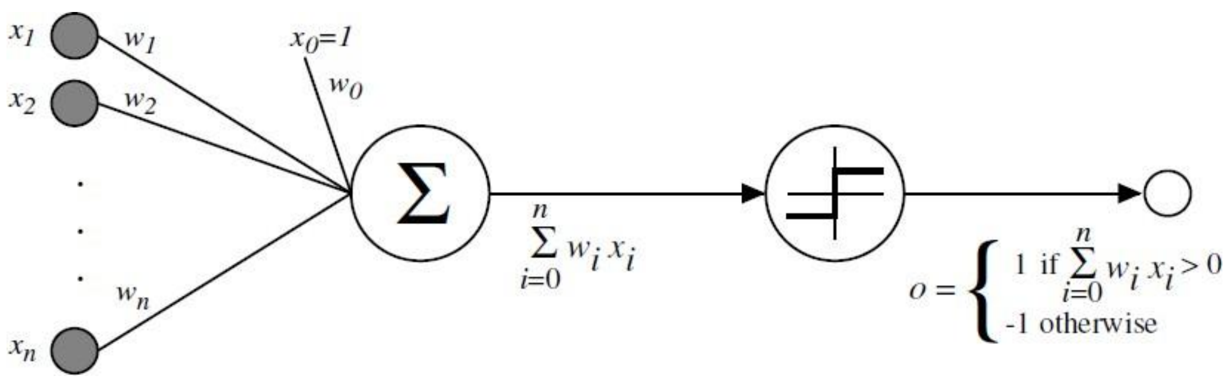
$$\frac{\partial E_d}{\partial net_j} = -(t_j - o_j)\ o_j(1 - o_j) \quad \text{.......equ(6)}$$

and combining this with Equations (1) and (2), we have the stochastic gradient descent rule for output units

$$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial w_{ji}} = \eta\ (t_j - o_j)\ o_j(1 - o_j)x_{ji} \quad \text{.......equ(7)}$$

14a) PERCEPTRON

- One type of ANN system is based on a unit called a perceptron. Perceptron is a single layer neural network.



- A perceptron takes a vector of real-valued inputs, calculates a linear combination of these inputs, then outputs a 1 if the result is greater than some threshold and -1 otherwise.
- Given inputs ***x*** through ***x***, the output **$O(x_1, \ldots, x_n)$** computed by the perceptron is

$$o(x_1, \ldots, x_n) = \begin{cases} 1 & \text{if } w_0 + w_1x_1 + \cdots + w_nx_n > 0 \\ -1 & \text{otherwise.} \end{cases}$$

- Where, each w_i is a real-valued constant, or weight, that determines the contribution of input x_i to the perceptron output.

- $-w_0$ is a threshold that the weighted combination of inputs $w_1x_1 + \ldots + w_nx_n$ must surpass in order for the perceptron to output a 1.

Sometimes, the perceptron function is written as,

$$O(\vec{x}) = \text{sgn}(\vec{w} \cdot \vec{x})$$

Where,

$$\text{sgn}(y) = \begin{cases} 1 & \text{if } y > 0 \\ -1 & \text{otherwise.} \end{cases}$$

Learning a perceptron involves choosing values for the weights $w_0, \ldots, w_n$. Therefore, the space H of candidate hypotheses considered in perceptron learning is the set of all possible real-valued weight vectors

$$H = \{\vec{w} \mid \vec{w} \in \Re^{(n+1)}\}$$

Representational Power of Perceptrons

- The perceptron can be viewed as representing a hyperplane decision surface in the n-dimensional space of instances (i.e., points)
- The perceptron outputs a 1 for instances lying on one side of the hyperplane and outputs a -1 for instances lying on the other side, as illustrated in below figure

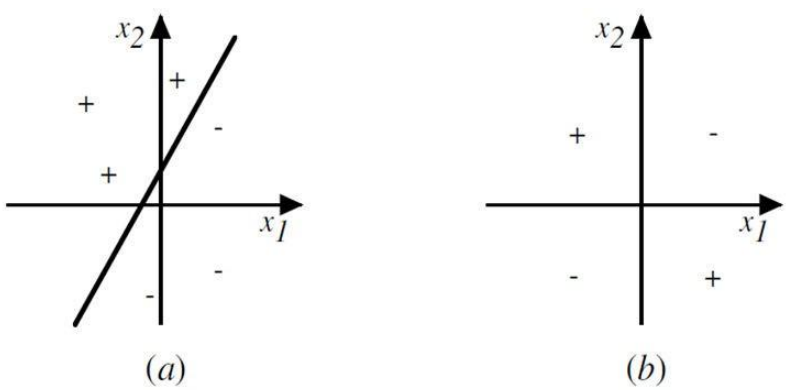


Figure : The decision surface represented by a two-input perceptron.
(a) A set of training examples and the decision surface of a perceptron that classifies them correctly. *(b)* A set of training examples that is not linearly separable.
x_1 and x_2 are the Perceptron inputs. Positive examples are indicated by "+", negative by "-".

The Perceptron Training Rule

The learning problem is to determine a weight vector that causes the perceptron to produce the correct + 1 or - 1 output for each of the given training examples.

<u>To learn an acceptable weight vector</u>

- Begin with random weights, then iteratively apply the perceptron to each training example, modifying the perceptron weights whenever it misclassifies an example.

- This process is repeated, iterating through the training examples as many times as needed until the perceptron classifies all training examples correctly.
- Weights are modified at each step according to the perceptron training rule, which revises the weight w_i associated with input xi according to the rule.

$$w_i \leftarrow w_i + \Delta w_i$$

Where,

$$\Delta w_i = \eta(t - o)x_i$$

Here,
t is the target output for the current training example
o is the output generated by the perceptron
η is a positive constant called the ***learning rate***

- The role of the learning rate is to moderate the degree to which weights are changed at each step. It is usually set to some small value (e.g., 0.1) and is sometimes made to decay as the number of weight-tuning iterations increases

---

14b) Perceptron A has weight values w0=1, w1=2, w2=1.
Perceptron B has weight values w0=0, w1=2, w2=1.
True or False? Perceptron A has more general than perceptron B.

Solution:

We will say that h_j is (strictly) more-general than h_k (written $h_j >_g h_k$) if and only if $(h_j \geq_g h_k) \wedge (h_k \ngeq_g h_j)$. Finally, we will sometimes find the inverse useful and will say that ***hj*** is ***more specific than*** h_k when h_k is ***more_general-than*** h_j.

X_1	X_2	$w_0+w_1x_1+w_2x_2$ Perceptron A	$w_0+w_1x_1+w_2x_2$ Perceptron B	A more general than B
0	0	1+2*0+1*0=1	0+2*0+1*0=0	1
0	1	1+2*0+1*1=2	0+2*0+1*1=1	1
1	0	1+2*1+1*0=3	0+2*1+1*0=2	1
1	1	1+2*1+1*1=4	0+2*1+1*1=3	1

B(<x1,x2>) = 1 2x1+x2 > 0 1+2x1+x2 > 0 A(<x1,x2>)) = 1

True.

Q)15)

Entropy(S) = 1

Entropy (STrue) = 0 bcz both belongs to –ve class

Entropy (No) = -(3/7) log(3/7) – (4/7) log(4/7)
= 0.9852

Hence

Info.gain(S, Refund) = Entropy(S) – [(3/10)*0 + (7/10) *0.9852]
= 0.8813 - [0 + (7/10)*0.9852]
= **0.19166**

Info.gain(S, Marital status) =
Entropy(S) – [(4/10)* Entropy (Single) + (4/10) *Entropy (Married) + (2/10)*Entropy (Divorced)]

Entropy (Single) = 1
Entropy (Married) = 0
Entropy (Divorced) = 1

Info.gain(S, Marital Status) = Entropy(S) – [(4/10)*(1) + (4/10) *(0) + ((2/10)*1]
= **0.2813**

Info.gain(S, Taxable Income) = Entropy(S) – [(3/10)* Entropy (<80k) + (7/10) *Entropy (>80k)]

Entropy (<80k) = 0
Entropy (>80k) = - (3/7) log (3/7) – (4/7) log (4/7)
= 0.9852

Info.gain(S, Taxable income) = Entropy(S) – [(3/10)*(0) + (7/10) *(0.9852)]
= **0.19166**

Among all marital status has highest information gain so it becomes the root node

Marital status=Single: Entropy (Single) =1
Gain (Single=Refund) = E (Single) – [(1/4)*E (Yes) + (3/4)*E (No)]
E (Yes) = 0
E (No) = (-2/3) log (2/3)-(1/3) log (1/3)
=0.9183
Gain (Single=Refund) = 1 – [(1/3)*0 + (3/4)*0.9183]
= 0.31127
Gain (Single=Taxable income) = E (Single) – [(1/4)*E (<80) + (3/4)*E (>80)]
=0.31127

Marital status = Divorced: Entropy (Divorced) =1
Gain (Divorced=Refund) = E (Divorced) – [(1/2)*E (Yes) + (1/2)*E (No)]
E (Yes) = 0
E (No) = 0
Gain = 1
Gain (Divorced=Taxable income) = E (Divorced) – [0+ (2/2)*E (>80k)] = 0