Q I

I) Understand the working of the following classifier algorithms and trace the same for a sample dataset (min 10 records) which involves 2 classes (Binary Classifier) eg: 'Yes' or 'No' , 'True' or 'False'.

a) Decision Tree Induction

Give pseudo code and trace decision tree algorithms.

Understand attribute selection measures such as Information gain, gain ratio (use anyone for the trace).

Pseudo Code

S – Samples, A – Attribute List

1. create a node N

2. If all samples are of the same class C then label N with C; Stop;

3. If A is empty then label N with the most common class C in S (majority voting); Stop;

4. Select a ∈ A, with the highest information gain; Label N with a;

5. For each value v of a:

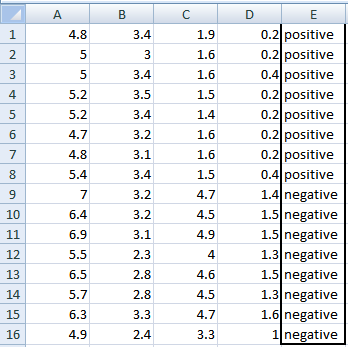
a. Grow a branch from N with condition a=y;

b. Let be the subset of samples in S with a=y;

c. If is empty then attach a leaf labelled with the most common class in S;

d. Else attach the node generated for Samples S, Attribute List A-a

Trace for Information Gain



A, B, C, D attributes can be considered as predictors

E column class labels can be considered as a target variable.

For constructing a decision tree from this data, we have to convert continuous data into categorical data.

Categorization is done by,

A: >=5 or <5

B: >=3 or <3

C: >=4.2 or <4.2

D: >=1.4 or <1.4

Entropy used for finding IG is given by,



Now calculating IG for A,

Var A has value >=5 for 12 records out of 16 and 4 records with value <5 value.

(A >= 5 and class is positive): 5/12

(A >= 5 and class is negative): 7/12

Entropy(5,7) = -1 \* ( (5/12)\*log2(5/12) + (7/12)\*log2(7/12)) = 0.9799

(A <5 and class is positive): ¾

(A <5 and class is negative): ¼

Entropy(3,1) =  -1 \* ( (3/4)\*log2(3/4) + (1/4)\*log2(1/4)) = 0.81128

Entropy(Target, A) = P(>=5) \* E(5,7) + P(<5) \* E(3,1)  
= (12/16) \* 0.9799 + (4/16) \* 0.81128 = 0.937745

Information Gain for A = E(Target) – E(Target, A) = 1 – 0.937745 = 0.062255

Similarly,

IG(A) = 0.062255

IG(B) = 0.707095

IG(C) = 0.5488

IG(D) = 0.41189

Using IG values in descending order we can construct DT as,

b) Naive Bayesian Classifier (NBC)

Read about Bayes theorem, conditional class independence, prior and posterior probabilities.

How to handle zero probability scenario (Laplacian Estimator)

Give a short pseudo code and trace.

Pseudo Code

Target Variable – Y (true or false -- binary)

Predictor Variables – X1 .. Xn

1. From the data, Estimate P(Y = true) and P(Y = false)
2. To do (1), For every value xij of each input attribute Xi,
   1. Find P(Xi = xij | Y = true)
   2. Find P(Xi = xij | Y = false)

Now, Classify a new X’ by,

(Let PROD(true) = {P(X1|Y=true). P(X2|Y=true)…P(Xn|Y=true)})

(Let PROD(false) = {P(X1|Y= false). P(X2|Y= false)…P(Xn|Y= false)})

Y’ (Predicted Class for X’) =

True if P(Y = true)PROD(true) > P(Y = false)PROD(false)

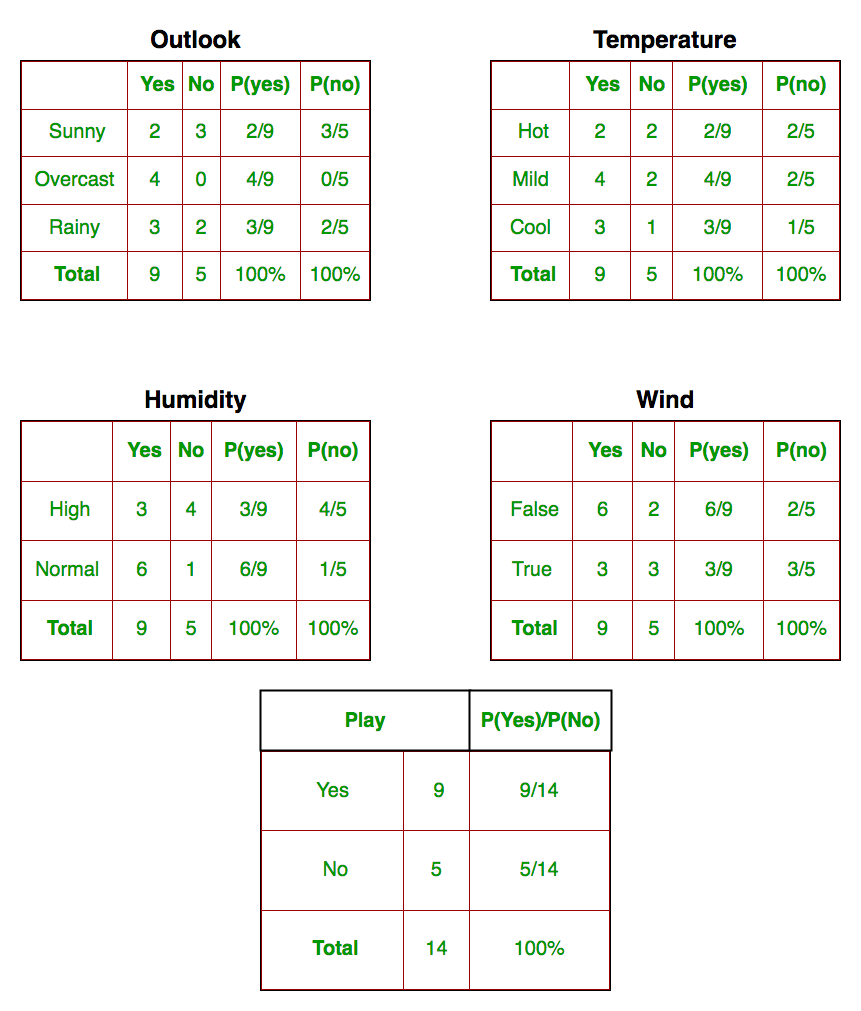
False if P(Y = true)PROD(true) <= P(Y = false)PROD(false)

Trace

| **INDEX** | **OUTLOOK** | **TEMPERATURE** | **HUMIDITY** | **WINDY** |
| --- | --- | --- | --- | --- |
| 0 | Rainy | Hot | High | False | No |
| 1 | Rainy | Hot | High | True | No |
| 2 | Overcast | Hot | High | False | Yes |
| 3 | Sunny | Mild | High | False | Yes |
| 4 | Sunny | Cool | Normal | False | Yes |
| 5 | Sunny | Cool | Normal | True | No |
| 6 | Overcast | Cool | Normal | True | Yes |
| 7 | Rainy | Mild | High | False | No |
| 8 | Rainy | Cool | Normal | False | Yes |
| 9 | Sunny | Mild | Normal | False | Yes |
| 10 | Rainy | Mild | Normal | True | Yes |
| 11 | Overcast | Mild | High | True | Yes |
| 12 | Overcast | Hot | Normal | False | Yes |
| 13 | Sunny | Mild | High | True | No |

We assume that all attributes are independent and are of equal importance to outcome.

Then we find P(Xi=xij|Y=Yes) and P(Xi=xij|Y=No)



Now that we have this data,

Suppose test data is

Today = (Sunny, Hot, Normal, False)

By Bayes Theorem,

P(Yes|today) =

= =

P(No|today) =

= =

As P(today) > 0,

Clearly P(Yes) > P(No)

So, predicted value for playing golf is YES.

For Problems with Zero Probability, use

Laplace smoothing: By applying this method, prior probability and conditional probability are, K is the number of different values in y and A is the number of different values in aj.

c) Neural Network Classifier (Back Propagation Network(BPN))

Understand basic terminologies - topology of a network, input layer, hidden layer, output layer, activation functions, weight, bias.

Strength and limitations of a neural network.

Take a sample network to learn the basics. For instance, learning the behaviour of OR gate, AND gate.

Take a network involving atleast one hidden layer and trace the BPN algorithm assuming a target class for one epoch.

Trace

Activation function: A = 1/(1 + e-x) (Sigmoid)

Function: y=2.x

Input output output before back propogation squared error

0 0 0 0

1 2 3 4

2 4 6 16

Derivative:

+ve : increase weight

-ve : decrease weight

0 : do nothing

Backpropogation starts: Decomposing the deriavtive to calculatable smaller derivatives= auto differnciation Rate 60x, loss=30units so reduce the weight by 0.5 New weight =old weight-derivative\*learning rate Iterate untill convergence h(X) = W0.X0 + W1.X1 + W2.X2 h(X) = sigma(W.X) for all (W, X) And X1 | X2 | Y 0 | 0 | 0 0 | 1 | 0 1 | 0 | 0 1 | 1 | 1 or X1 | X2 | Y 0 | 0 | 0 0 | 1 | 1 1 | 0 | 1 1 | 1 | 1 Unit Z1: h(x) = W0.X0 + W1.X1 + W2.X2= 1 . 0 + 1 . 0 + 1 . 0 = 1 =a z = f(a) = a => z = f(1) = 1 delta\_0 = w . delta\_1 . f'(z) W := W - alpha . J'(W) J'(W) = Z . delta Consider exor gate for calculation nit Z2: h(x) = W0.X0 + W1.X1 + W2.X2 = 1 . 1 + 1 . 0 + 1 . 0= 1 = a z = f(a) = a => z = f(1) = 1 Unit Z3: h(x) = W0.X0 + W1.X1 + W2.X2 = 1 . 1 + 1 . 0 + 1 . 0= 1 = z = f(a) = a => z = f(1) = 1 Unit D0: h(x) = W0.Z0 + W1.Z1 + W2.Z2 + W3.Z3 = 1 . 1 + 1 . 1 + 1 . 1 + 1 . 1= 4 = z = f(a) = a => z = f(4) = 4 delta\_D0 = total\_loss = -4 W11 := 1.4 W21 := 1.4 W31 := 1.4 after calculation from the above formula V01 := 1.4 V02 := 1.4 V03 := 1.4

d) SVM or Nearest Neighbour Classifier (Anyone)

Understand the algorithm/working and give the pseudocode and the trace

Pseudo Code

k-Nearest Neighbour

Classify (X, Y, x) where X: training data, Y: class labels of X, x: test sample

For i = 1 to m do

Computer distance d(Xi, x)

End for

Computer set I containing indices for the k smallest distances d(Xi, x)

Return majority label for {Yi where i ∈ I}

Trace

Consider binary Y with attributes X

**K = 3**

|  |  |  |
| --- | --- | --- |
| X1 | X2 | Y |
| 1 | 2 | False |
| 2 | 3 | True |
| 3 | 1 | True |
| 4 | 3 | False |
| 5 | 5 | False |
| 6 | 1 | False |
| 7 | 2 | True |

Now for a test x = (2.5, 3.5),

D(X, x) =>

|  |  |  |
| --- | --- | --- |
| X1 | X2 | D(X, x) |
| 1 | 2 | 2.12 |
| 2 | 3 | 0.71 |
| 3 | 1 | 2.55 |
| 4 | 3 | 1.58 |
| 5 | 5 | 2.91 |
| 6 | 1 | 4.30 |
| 7 | 2 | 4.74 |

As K = 3, minimum 3 distances is for (2, 3), (4, 3), (1, 2)

And their Y is True, False, False

SO, majority is FALSE = y for x