Sivaranjani Prabasankar A20436206

Homework 3

1. Build a decision Tree [50]

Color	Size	Act	Age	Inflated
YELLOW	SMALL	STRETCH	Young	T
YELLOW	SMALL	STRETCH	Old	T
YELLOW	SMALL	STRETCH	Old	T
YELLOW	SMALL	DIP	Kid	F
YELLOW	SMALL	DIP	Kid	T
YELLOW	LARGE	STRETCH	Old	T
YELLOW	LARGE	STRETCH	Old	T
YELLOW	LARGE	DIP	Young	T
YELLOW	LARGE	DIP	Young	T
YELLOW	LARGE	DIP	Young	F
PURPLE	SMALL	STRETCH	Young	F
PURPLE	SMALL	STRETCH	Old	T
PURPLE	SMALL	STRETCH	Old	F
PURPLE	SMALL	DIP	Kid	T
PURPLE	SMALL	DIP	Kid	F
Blue	LARGE	STRETCH	Kid	T
Blue	LARGE	DIP	Young	F
Blue	LARGE	DIP	Young	F
Blue	LARGE	DIP	Old	T
Blue	LARGE	DIP	Young	T

Aim: To classify whether a balloon is inflated or not.

Method: The decision tree is built in a top-down fashion using features. The feature should be the best splits the target class into the purest possible children nodes. Feature selection is the key component in decision trees, we want to predict what features of the data are relevant to the target class.

I [13,7]

Sivaranjani Prabasankar A20436206

Level 1: Selecting Root node

Gain = Entropy before split – Entropy after split

STEP 1.1: Calculate Entropy before split for class Inflated.

Entropy = I (p,n) = – Pr(P) $\log_2$ Pr(P) – Pr(N) $\log_2$ Pr(N)

Pr(P) = p / (p+n) **Pr(N) = n / (p+n)**

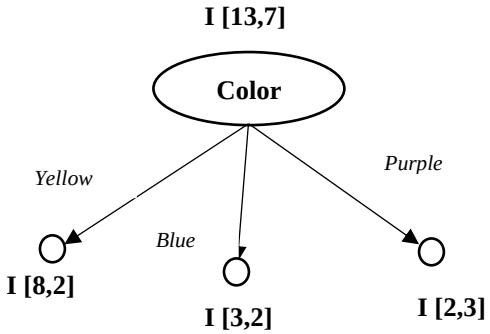
For Inflated, the labels are
No. of. True = p = 13
No. of. false = n = 7

I (13,7) = – {(13/20) * $\log_2$ (13/20)} – {(7/20) * $\log_2$ (7/20)}
= – (0.65) * $\log_2$ (0.65) – (0.35) * $\log_2$ (0.35)
= 0.403967445 + 0.53010061

I (13,7) = 0.908411596868095

Entropy before split = 0.908

STEP 1.2: Calculate Entropy after split for class Color.



Entropy = I (p,n) = – Pr(P) $\log_2$ Pr(P) – Pr(N) $\log_2$ Pr(N)

Pr(P) = p / (p+n) **Pr(N) = n / (p+n)**

No. of. True = p, No. of. false = n

For Color = Yellow, the labels are

No. of. True with Color = Yellow = p = 8, No. of. False with Color = Yellow = n = 2

I (8,2) = – {(8/10) * $\log_2$ (8/10)} – {(2/10) * $\log_2$ (2/10)} = 0.721928095

Sivaranjani Prabasankar A20436206

For Color = Blue, the labels are

No. of. True with Color = Blue = p = 3, No. of. False with Color = Blue = n = 2

$I(3,2) = -\{(3/5) * \log_2(3/5)\} - \{(2/5) * \log_2(2/5)\} = 0.970950594$

For Color = Purple, the labels are

No. of. True with Color = Purple = p = 2, No. of. False with Color = Purple = n = 3

$I(2,3) = -\{(2/5) * \log_2(2/5)\} - \{(3/5) * \log_2(3/5)\} = 0.970950594$

Entropy after split of Color

= (10/20) * I (8,2) + (5/20) * I (3,2) + (5/20) * I (2,3)
= 0.5 * 0.722 + 0.25* 0.970+ 0.25* 0.970
= 0.361 + 0.2425 + 0.2425 = 0.846

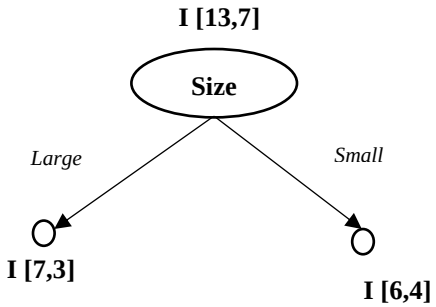
Entropy after split= 0.846

Information Gain = Entropy before split – Entropy after split

Gain $_{(Color)}$ = 0.934 - 0.846
= 0.088

⇨ **Information Gain if we split using Color label is 0.088**

STEP 1.3: Calculate Entropy after split for class Size.



Entropy = I (p,n) = $-\Pr(P)\log_2\Pr(P) - \Pr(N)\log_2\Pr(N)$

Pr(P) = p / (p+n) **Pr(N) = n / (p+n)**

No. of. True = p, No. of. false = n

For Size = Large, the labels are

No. of. True with Size = Large = p = 7, No. of. False with Size = Large = n = 3

Sivaranjani Prabasankar A20436206

I (7,3) = – {(7/10) * $\log_2$ (7/10)} – {(3/10) * $\log_2$ (3/10)} = 0.881290899

For Size = Small, the labels are

No. of. True with Size = Small = p = 7, No. of. False with Size = Small = n = 3

I (6,4) = – {(6/10) * $\log_2$ (6/10)} – {(4/10) * $\log_2$ (4/10)} = 0.970950594

Entropy after split of Size

= (10/20) * I (7,3) + (10/20) * I (6,4)
= 0.5 * 0.881 + 0.50* 0.970
= 0.4405 + 0.485= 0.9255

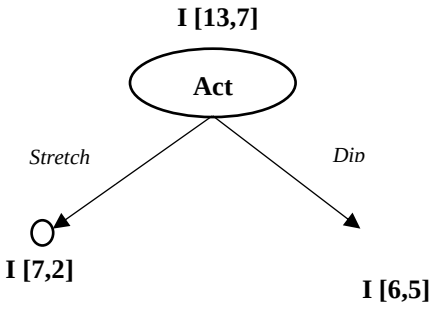
Entropy after split= 0.9255

Information Gain = Entropy before split – Entropy after split

Gain $_{(Size)}$ = 0.934 - 0. 9255
= 0.009

⇨ **Information Gain if we split using Size label is 0.009**

STEP 1.4: Calculate Entropy after split for class Act.



Entropy = I (p,n) = – Pr(P) $\log_2$ Pr(P) – Pr(N) $\log_2$ Pr(N)

Pr(P) = p / (p+n) **Pr(N) = n / (p+n)**

No. of. True = p, No. of. false = n

For Act = Stretch, the labels are

No. of. True with Act = Stretch = p = 7, No. of. False with Act = Stretch = n = 2

I (7,2) = – {(7/9) * $\log_2$ (7/9)} – {(2/9) * $\log_2$ (2/9)} = 0.764204507

Sivaranjani Prabasankar A20436206

For Act = Dip, the labels are

No. of. True with Act = Dip = p = 6, No. of. False with Act = Dip = n = 5

I (6,5) = – {(6/11) * $\log_2$ (6/11)} – {(5/11) * $\log_2$ (5/11)} = 0.994030211

Entropy after split of Act

= (9/20) * I (7,2) + (11/20) * I (6,5)
= 0.45 * 0.764204507 + 0.55* 0.994030211
= 0.344+ 0.547 = 0.891

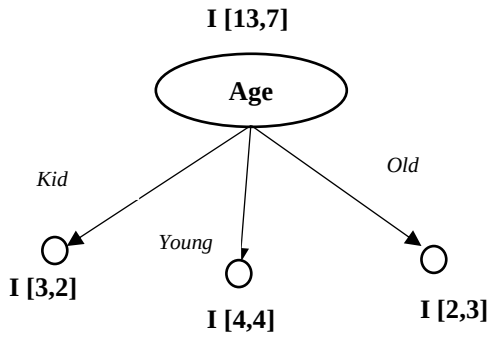
Entropy after split= 0.891

Information Gain = Entropy before split – Entropy after split

Gain $_{(Act)}$ = 0.934 - 0.891
= 0.043

⇨ **Information Gain if we split using Act label is 0.043**

STEP 1.5: Calculate Entropy after split for class Age.



Entropy = I (p,n) = – Pr(P) $\log_2$ Pr(P) – Pr(N) $\log_2$ Pr(N)

Pr(P) = p / (p+n) **Pr(N) = n / (p+n)**

No. of. True = p, No. of. false = n

For Age = Kid, the labels are

No. of. True with Age = Kid = p = 3, No. of. False with Age = Kid = n = 2

I (3,2) = – {(3/5) * $\log_2$ (3/5)} – {(2/5) * $\log_2$ (2/5)} = 0.970950594

Sivaranjani Prabasankar A20436206

For Age = Young, the labels are

No. of. True with Age = Young = p = 4, No. of. False with Age = Young = n = 4

$I(4,4) = -\{(4/8) * \log_2 (4/8)\} - \{(4/8) * \log_2 (4/8)\} = 0.591672779$

For Age = Old, the labels are

No. of. True with Age = Old = p = 2, No. of. False with Age = Old = n = 3

$I(6,1) = -\{(6/7) * \log_2 (6/7)\} - \{(1/7) * \log_2 (1/7)\} = 1$

Entropy after split of Age

$= (10/20) * I(3,2) + (5/20) * I(4,4) + (5/20) * I(6,1)$
$= 0.25 * 0.971 + 0.35 * 0.592 + 0.4 * 1$
$= 0.24275 + 0.2072 + 0.4 = 0.84995$

Entropy after split= 0.84995

Information Gain = Entropy before split – Entropy after split

$Gain_{(Age)} = 0.934 - 0.84995$
$= 0.084$

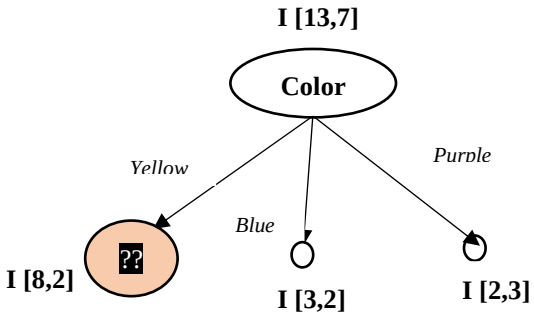
⇨ **Information Gain if we split using Age label is 0.084**

Label	Information Gain
Color	0.088
Size	0.009
Act	0.043
Age	0.084

Decision: From above values, we can witness that Color label has highest information gain than others. Hence, we can proceed with Color as Root node.

Sivaranjani Prabasankar A20436206

Level 2: Selecting Sub-Root node



Color	Size	Act	Age	Inflated
YELLOW	SMALL	STRETCH	Young	T
YELLOW	SMALL	STRETCH	Old	T
YELLOW	SMALL	STRETCH	Old	T
YELLOW	SMALL	DIP	Kid	F
YELLOW	SMALL	DIP	Kid	T
YELLOW	LARGE	STRETCH	Old	T
YELLOW	LARGE	STRETCH	Old	T
YELLOW	LARGE	DIP	Young	T
YELLOW	LARGE	DIP	Young	T
YELLOW	LARGE	DIP	Young	F

Selecting Level 2 nodes

Gain = Entropy before split – Entropy after split

STEP 2.1: Calculate Entropy before split for class Inflated when Color = Yellow.

Entropy = I (p,n) = – Pr(P) $\log_2$ Pr(P) – Pr(N) $\log_2$ Pr(N)

Pr(P) = p / (p+n) **Pr(N) = n / (p+n)**

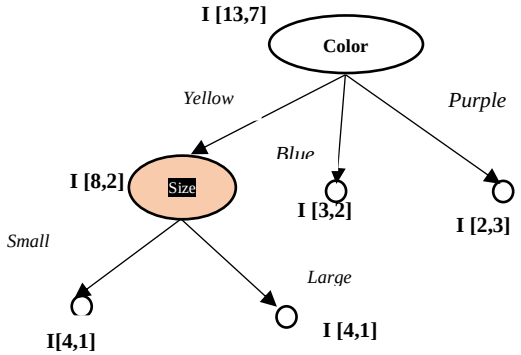
For Inflated, the labels are
No. of. True = p = 8
No. of. false = n = 2

$I_{(Yellow)}$ = – {(8/10) * $\log_2$ (8/10)} – {(2/10) * $\log_2$ (2/10)}
= – (0.8) * $\log_2$ (0.8) – (0.2) * $\log_2$ (0.2)
= 0.07752801 + 0.019382003

I_{Yellow} (8,2) = 0.721928095
Entropy before split = 0.722

Sivaranjani Prabasankar A20436206

STEP 2.2: Calculate Entropy after split for class Size when Color = Yellow.



Entropy = I (p,n) = – Pr(P) $\log_2$ Pr(P) – Pr(N) $\log_2$ Pr(N)
Pr(P) = p / (p+n) **Pr(N) = n / (p+n)**

No. of. True = p, No. of. false = n

For Size = Large and Color = Yellow, the labels are

No. of. True with Size = Large & Color = Yellow = p = 4,
No. of. False with Size = Large & Color = Yellow = n = 1

I (4,1) = – {(4/5) * $\log_2$ (4/5)} – {(1/5) * $\log_2$ (1/5)}
= 0.257542476 +0.464385619
= 0.721928095

For Size = Small and Color = Yellow, the labels are

No. of. True with Size = Small & Color = Yellow = p = 4,
No. of. False with Size = Small & Color = Yellow = n = 1

I (4,1) = – {(4/5) * $\log_2$ (4/5)} – {(1/5) * $\log_2$ (1/5)}
= 0.257542476 +0.464385619
= 0.721928095

Entropy after split of Size

$I_{(Yellow, Size)}$ = (5/10) * I (4,1) + (5/10) * I (4,1)
= 0.5 * 0.722 + 0.50* 0.722
= 0.361 + 0.361= 0.722

Entropy after split= 0.722

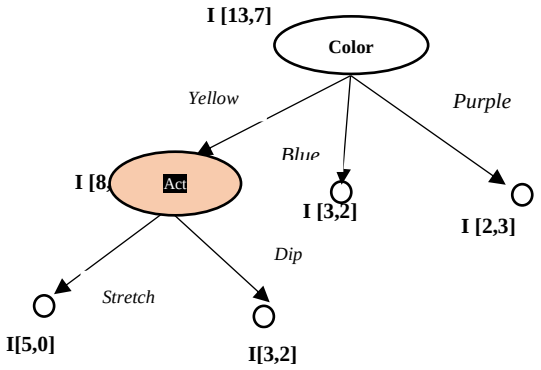
Information Gain = Entropy before split – Entropy after split

Information Gain $_{(Yellow, Size)}$ = **$I_{(Yellow)} - I_{(Yellow, Size)}$**
= 0.722 - 0. 722 = 0

⇨ **Information Gain if we split using Size label is 0.0**

Sivaranjani Prabasankar A20436206

STEP 2.3: Calculate Entropy after split for class Act when Color = Yellow.



Entropy = I (p,n) = – Pr(P) $\log_2$ Pr(P) – Pr(N) $\log_2$ Pr(N)

Pr(P) = p / (p+n) **Pr(N) = n / (p+n)**

No. of. True = p, No. of. false = n

For Act = Stretch & Color = Yellow, the labels are

No. of. True with Act = Stretch and Color = Yellow = p = 5,
No. of. False with Act = Stretch and Color = Yellow = n = 0

I (5,0) = – {(5/5) * $\log_2$ (5/5)} – {(0/5) * $\log_2$ (0/5)} = 0

For Act = Dip & Color = Yellow, the labels are

No. of. True with Act = Dip = p = 3, No. of. False with Act = Dip = n = 2

I (3,2) = – {(3/5) * $\log_2$ (3/5)} – {(2/5) * $\log_2$ (2/5)}
= 0.442179356 + 0.528771238
= 0.970950594

Entropy after split of Act

$I_{(Yellow, Act)}$ = (5/10) * I (5,0) + (5/10) * I (3,2)
= 0.5 * 0.0 + 0.5* 0. 970950594
= 0+ 0.485475297
= 0.485475297

Entropy after split= 0.485

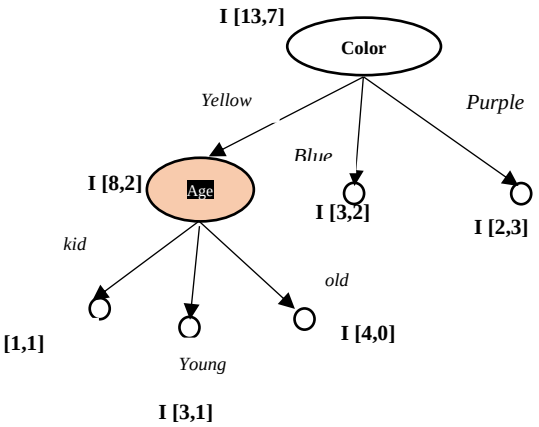
Information Gain = Entropy before split – Entropy after split

Information Gain $_{(Yellow, Act)}$ = $I_{(Yellow)}$ - $I_{(Yellow, Act)}$
= 0.722 - 0.485 = 0.237

⇨ **Information Gain if we split using Act label is 0.237**

Sivaranjani Prabasankar A20436206

STEP 2.4: Calculate Entropy after split for class Age when Color = Yellow.



Entropy = I (p,n) = – Pr(P) $\log_2$ Pr(P) – Pr(N) $\log_2$ Pr(N)

Pr(P) = p / (p+n) **Pr(N) = n / (p+n)**

No. of. True = p, No. of. false = n

For Age = Kid and Color = Yellow, the labels are

No. of. True with Age = Kid and Color = Yellow = p = 1,
No. of. False with Age = Kid and Color = Yellow = n = 2

I (1,1) = – {(1/2) * $\log_2$ (1/2)} – {(1/2) * $\log_2$ (1/2)}
= 0.5 + 0.5
= 1

For Age = Young and Color = Yellow, the labels are

No. of. True with Age = Young and Color = Yellow = p = 3,
No. of. False with Age = Young and Color = Yellow = n = 1

I (3,1) = – {(3/4) * $\log_2$ (3/4)} – {(1/4) * $\log_2$ (1/4)}
= 0.311278124 + 0.5
= 0.811278124

For Age = Old and Color = Yellow, the labels are

No. of. True with Age = Old and Color = Yellow = p = 4,
No. of. False with Age = Old and Color = Yellow = n = 0

I (4,0) = – {(4/4) * $\log_2$ (4/4)} – {(0/4) * $\log_2$ (0/4)} = 0

Entropy after split of Age

Sivaranjani Prabasankar A20436206

$I_{(Yellow, Age)}$ = (2/10) * I (1,1) + (4/10) * I (3,1) + (4/10) * I (4,0)
= 0.2 * 1 + 0.4* 0.811278124 + 0.4* 0
= 0.2 + 0.3245112496 + 0
= 0.5245112496

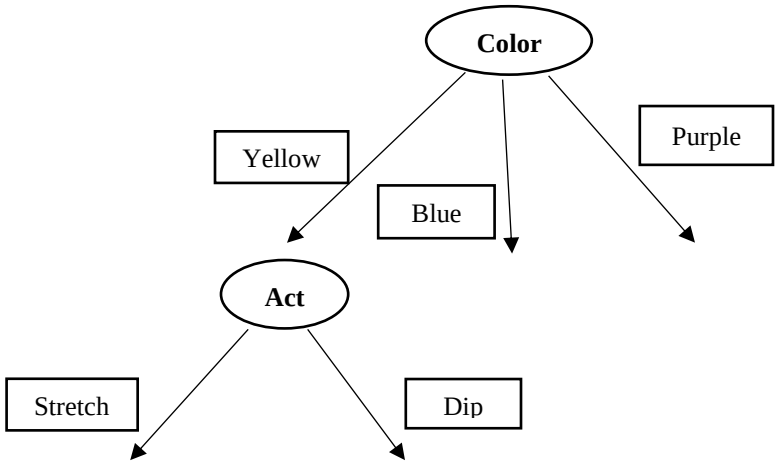
Entropy after split= 0.5245
Information Gain = Entropy before split – Entropy after split

Information Gain $_{(Yellow, Age)}$ = $I_{(Yellow)} - I_{(Yellow, Age)}$
= 0.722 - 0.5245
= 0.1975

Information Gain if we split using Age label is 0.1975

Label	Information Gain when Color = Yellow
Size	0.0
Act	0.237
Age	0.1975

From above values, we can witness that Act label has highest information gain than others when Color = Yellow. Hence, we can proceed with Act as Root node after the Root Node Split (Color).



Color	Size	Act	Age	Inflated
Blue	LARGE	STRETCH	Kid	T
Blue	LARGE	DIP	Young	F
Blue	LARGE	DIP	Young	F
Blue	LARGE	DIP	Old	T
Blue	LARGE	DIP	Young	T

Sivaranjani Prabasankar A20436206

Selecting Level 2 nodes

Gain = Entropy before split – Entropy after split

STEP 2.1: Calculate Entropy before split for class Inflated when Color = Blue.

Entropy = I (p,n) = – Pr(P) $\log_2$ Pr(P) – Pr(N) $\log_2$ Pr(N)

Pr(P) = p / (p+n) **Pr(N) = n / (p+n)**

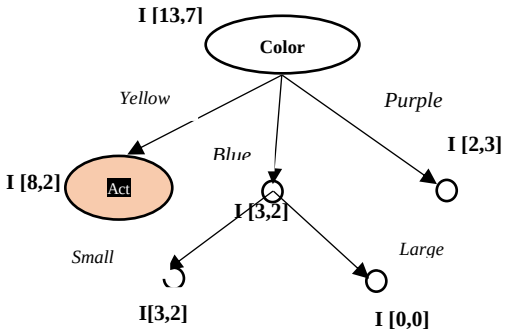
For Inflated, the labels are
No. of. True = p = 3
No. of. false = n = 2

I_{Blue} = – {(3/5) * $\log_2$ (3/5)} – {(2/5) * $\log_2$ (2/5)}
= 0.442179356 + 0.528771238
= 0.970950594

I_{Blue} = 0.970950594

Entropy before split = 0.971

STEP 2.2: Calculate Entropy after split for class Size when Color = Blue.



Entropy = I (p,n) = – Pr(P) $\log_2$ Pr(P) – Pr(N) $\log_2$ Pr(N)

Pr(P) = p / (p+n) **Pr(N) = n / (p+n)**

No. of. True = p, No. of. false = n

For Size = Large and Color = Blue, the labels are

No. of. True with Size = Large & Color = Blue = p = 3,
No. of. False with Size = Large & Color = Blue = n = 2

I (3,2) = – {(3/5) * log2 (3/5)} – {(2/5) * log2 (2/5)}

Sivaranjani Prabasankar A20436206

= 0.442179356+ 0.528771238
= 0.970950594

For Size = Small and Color = Blue, the labels are

No. of. True with Size = Small & Color = Blue = p = 0,
No. of. False with Size = Small & Color = Blue = n = 0

I (0,0) = 0

Entropy after split of Size

$I_{(Blue, Size)}$ = (5/5) * I (3,2) + (0/10) * I (0,0)
= 1 * 0.970950594 + 0
= 0.970950594

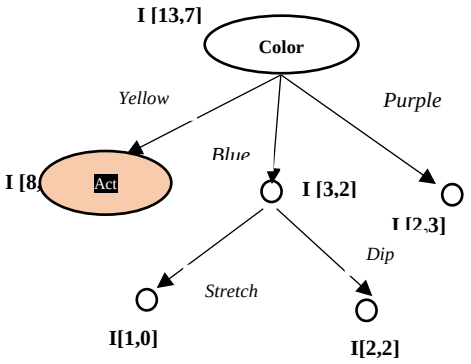
Entropy after split= 0.971

Information Gain = Entropy before split – Entropy after split

Information Gain (C_{Blue}, Size) = $I_{(Blue)} - I_{(Blue, Size)}$
= 0.971 - 0. 971
= 0

Information Gain if we split using Size label is 0.0 when Color is Blue

STEP 2.3: Calculate Entropy after split for class Act when Color = Blue.



Entropy = I (p,n) = – Pr(P) $\log_2$ Pr(P) – Pr(N) $\log_2$ Pr(N)

Pr(P) = p / (p+n) **Pr(N) = n / (p+n)**

No. of. True = p, No. of. false = n
For Act = Stretch & Color = Blue, the labels are

Sivaranjani Prabasankar A20436206

No. of. True with Act = Stretch and Color = Blue = p = 1,
No. of. False with Act = Stretch and Color = Blue = n = 0

$I(1,0) = -\{(1/1) * \log_2(1/1)\} - \{(0/1) * \log_2(0/1)\} = 0$

For Act = Dip & Color = Blue, the labels are
No. of. True with Act = Dip and Color = Blue = p = 2,
No. of. False with Act = Dip and Color = Blue = n = 2

$I(2,2) = -\{(2/4) * \log_2(2/4)\} - \{(2/4) * \log_2(2/4)\}$
$= 0.5 + 0.5$
$= 1$

Entropy after split of Act

$I_{(Blue, Act)} = (1/5) * I(1,0) + (4/5) * I(2,2)$
$= 0.2 * 0.0 + 0.8 * 1$
$= 0.8$

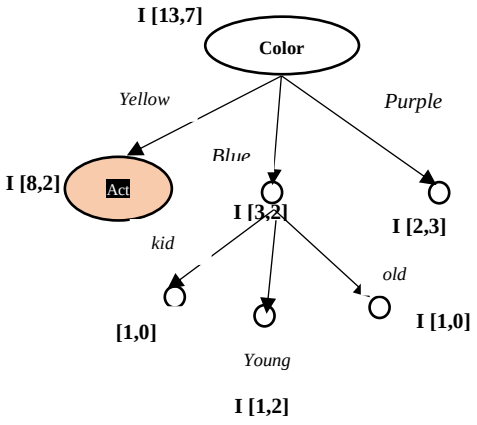
Entropy after split= 0.8

Information Gain = Entropy before split – Entropy after split

Information Gain $(C_{Blue}, Act) = I_{(Blue)} - I_{(Blue, Act)}$
$= 0.971 - 0.8$
$= 0.171$

Information Gain if we split using Act label is 0.171 when Color is Blue

STEP 2.4: Calculate Entropy after split for class Age when Color = Blue.



Entropy = I (p,n) = $-\Pr(P)\log_2\Pr(P) - \Pr(N)\log_2\Pr(N)$

$\Pr(P) = p/(p+n)$ **$\Pr(N) = n/(p+n)$**

Sivaranjani Prabasankar A20436206

No. of. True = p, No. of. false = n

For Age = Kid and Color = Blue, the labels are

No. of. True with Age = Kid and Color = Blue = p = 1,
No. of. False with Age = Kid and Color = Blue = n = 0

$I(1,0) = -\{(1/1) * \log_2(1/1)\} - \{(0/1) * \log_2(0/1)\} = 0$

For Age = Young and Color = Blue, the labels are

No. of. True with Age = Young and Color = Blue = p = 1,
No. of. False with Age = Young and Color = Blue = n = 2

$I(1,2) = -\{(1/3) * \log_2(1/3)\} - \{(2/3) * \log_2(2/3)\}$
$= 0.528320834 + 0.389975$
$= 0.918295834$

For Age = Old and Color = Blue, the labels are

No. of. True with Age = Old and Color = Blue = p = 1,
No. of. False with Age = Old and Color = Blue = n = 0

$I(1,0) = -\{(1/1) * \log_2(1/1)\} - \{(0/1) * \log_2(0/1)\} = 0$

Entropy after split of Age

$= (1/5) * I(1,0) + (3/5) * I(1,2) + (1/5) * I(1,0)$
$= 0.2 * 0 + 0.6 * 0.918295834 + 0.2 * 0$
$= 0.5509775004$

Entropy after split= 0.551
Information Gain = Entropy before split – Entropy after split

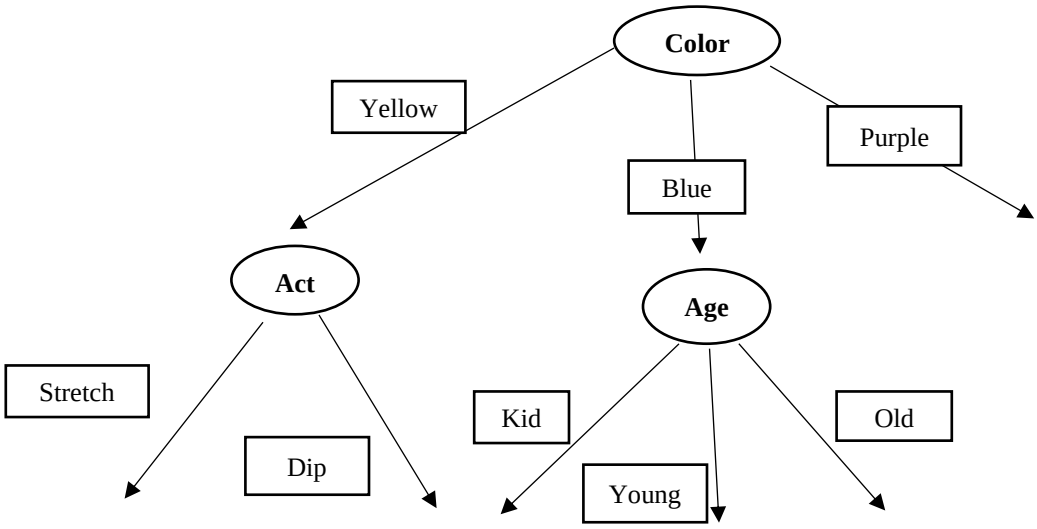
Information Gain $(C_{Blue}, Age) = \mathbf{I_{(Blue)} - I_{(Blue, Age)}}$
$= 0.971 - 0.551$
$= 0.42$

Information Gain if we split using Age label is 0.42 when color is Blue

Label	Information Gain when Color = Blue
Size	0.0
Act	0.171
Age	0.42

Sivaranjani Prabasankar A20436206

From above values, we can witness that Age label has highest information gain than others when Color = Blue. Hence, we can proceed with Age as Root node after the Root Node Split (Color).



Color	Size	Act	Age	Inflated
PURPLE	SMALL	STRETCH	Young	F
PURPLE	SMALL	STRETCH	Old	T
PURPLE	SMALL	STRETCH	Old	F
PURPLE	SMALL	DIP	Kid	T
PURPLE	SMALL	DIP	Kid	F

Selecting Level 2 nodes

Gain = Entropy before split – Entropy after split

STEP 2.1: Calculate Entropy before split for class Inflated when Color = Purple.

Entropy = I (p,n) = – Pr(P) $\log_2$ Pr(P) – Pr(N) $\log_2$ Pr(N)

Pr(P) = p / (p+n) **Pr(N) = n / (p+n)**

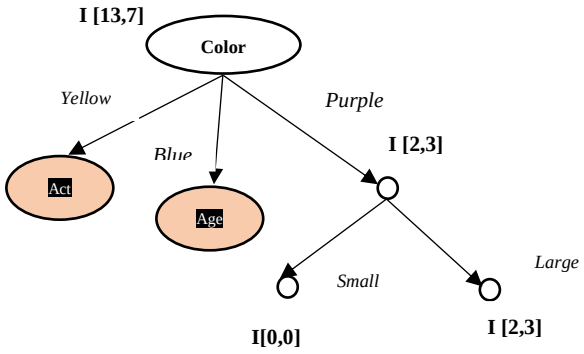
For Inflated, the labels are
No. of. True = p = 2
No. of. false = n = 3

I_{Purple} (2,3) = {(2/5) * $\log_2$ (2/5)} – {(3/5) * $\log_2$ (3/5)}
= 0.528771238 + 0.442179356
= 0.970950594

Sivaranjani Prabasankar A20436206

I_{Purple} (2,3) = 0.970950594
Entropy before split = 0.971

STEP 2.2: Calculate Entropy after split for class Size when Color = Purple.



Entropy = I (p,n) = – Pr(P) $\log_2$ Pr(P) – Pr(N) $\log_2$ Pr(N)

Pr(P) = p / (p+n) **Pr(N) = n / (p+n)**

No. of. True = p, No. of. false = n

For Size = Large and Color = Purple, the labels are

No. of. True with Size = Large & Color = Purple = p = 0,
No. of. False with Size = Small & Color = Purple = n = 0

I (0,0) = 0

For Size = Small and Color = Purple, the labels are

No. of. True with Size = Small & Color = Purple = p = 3,
No. of. False with Size = Small & Color = Purple = n = 2

I (2,3) = – {(2/5) * log2 (2/5)} – {(3/5) * log2 (3/5)}
= 0.528771238 + 0.442179356
= 0.970950594

Entropy after split of Size

$I_{(Blue, Size)}$ = (0/5) * I (0,0) + (5/5) * I (3,2)
= 0 + 1 * 0.970950594
= 0.970950594
Entropy after split= 0.971

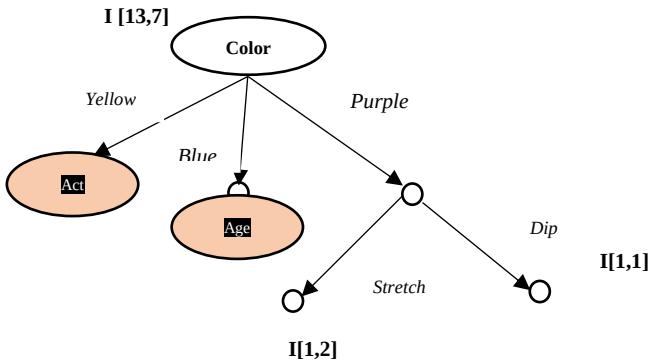
Sivaranjani Prabasankar A20436206

Information Gain = Entropy before split – Entropy after split

Information Gain (C_{Blue}, Size) = **$I_{(Blue)}$ - $I_{(Blue, Size)}$**
= 0.971 - 0. 971
= 0

⇨ **Information Gain if we split using Size label is 0.0 when Color is Purple**

STEP 2.3: Calculate Entropy after split for class Act when Color = Purple.



Entropy = I (p,n) = – Pr(P) $\log_2$ Pr(P) – Pr(N) $\log_2$ Pr(N)

Pr(P) = p / (p+n) **Pr(N) = n / (p+n)**

No. of. True = p, No. of. false = n

For Act = Stretch & Color = Purple, the labels are

No. of. True with Act = Stretch and Color = Purple = p = 1,
No. of. False with Act = Stretch and Color = Purple = n = 2

I (1,2) = – {(1/3) * $\log_2$ (1/3)} – {(2/3) * $\log_2$ (2/3)}
= 0.528320834 + 0.389975
= 0.918295834

For Act = Dip & Color = Purple, the labels are

No. of. True with Act = Dip and Color = Purple = p = 1,
No. of. False with Act = Dip and Color = Purple = n = 1

I (1,1) = – {(1/2) * $\log_2$ (1/2)} – {(1/2) * $\log_2$ (1/2)}
= 0.5 + 0.5
= 1

Entropy after split of Act

Sivaranjani Prabasankar A20436206

$I_{(Blue, Act)}$ = (3/5) * I (1,2) + (2/5) * I (1,1)
= 0.6 * 0.918295834 + 0.4* 1
= 0.5509775004 + 0.4
= 0.9509775004

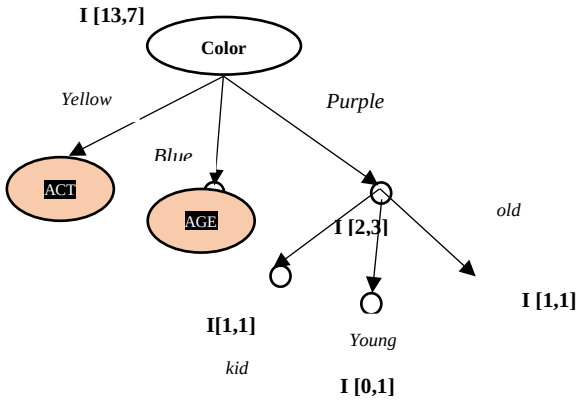
Entropy after split= 0.951

Information Gain = Entropy before split – Entropy after split

Information Gain (C_{Blue}, act) = $I_{(Blue)} - I_{(Blue, Act)}$
= 0.971 - 0.951
= 0.02

Information Gain if we split using Act label is 0.02 when Color is Purple

STEP 2.4: Calculate Entropy after split for class Age when Color = Purple.



Entropy = I (p,n) = – Pr(P) $\log_2$ Pr(P) – Pr(N) $\log_2$ Pr(N)

Pr(P) = p / (p+n) **Pr(N) = n / (p+n)**

No. of. True = p, No. of. false = n

For Age = Kid and Color = Purple, the labels are

No. of. True with Age = Kid and Color = Purple = p = 1,
No. of. False with Age = Kid and Color = Purple = n = 1

I (1,1) = – {(1/2) * $\log_2$ (1/2)} – {(1/2) * $\log_2$ (1/2)}
= 0.5 + 0.5
= 1

For Age = Young and Color = Purple, the labels are
No. of. True with Age = Young and Color = Purple = p = 0,

Sivaranjani Prabasankar A20436206

No. of. False with Age = Young and Color = Purple = n = 1

$I(0,1) = -\{(0/1) * \log_2(0/1)\} - \{(1/1) * \log_2(1/1)\} = 0$

For Age = Old and Color = Purple, the labels are

No. of. True with Age = Old and Color = Purple = p = 1,
No. of. False with Age = Old and Color = Purple = n = 1

$I(1,1) = -\{(1/2) * \log_2(1/2)\} - \{(1/2) * \log_2(1/2)\}$
$= 0.5 + 0.5$
$= 1$

Entropy after split of Age

$\mathbf{I}_{(Blue, Age)} = (2/5) * I(1,1) + (1/5) * I(1,0) + (2/5) * I(1,1)$
$= (0.4 * 1) + (0.2 * 0) + (0.4 * 1)$
$= 0.4 + 0.4$
$= 0.8$

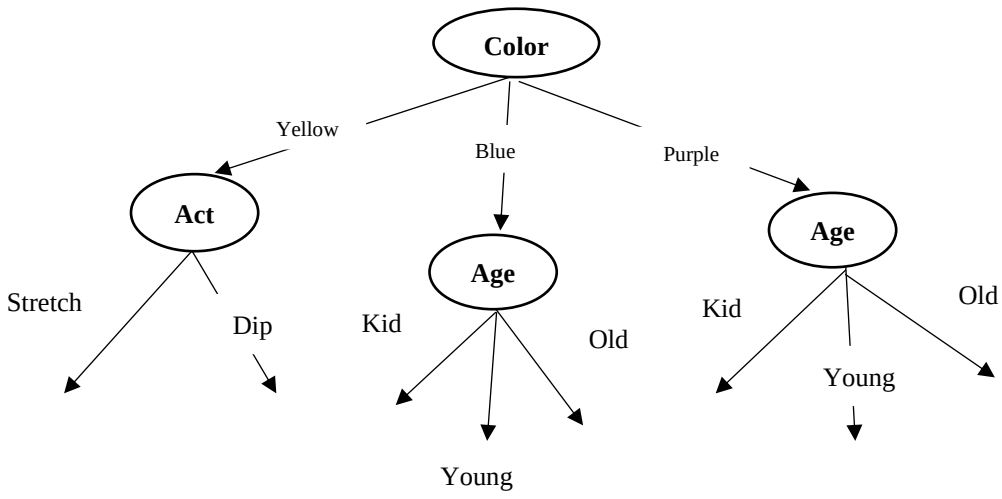
Entropy after split= 0.8
Information Gain = Entropy before split – Entropy after split

Information Gain $(C_{Blue, Age})$ $= \mathbf{I}_{(Blue)} - \mathbf{I}_{(Blue, Age)}$
$= 0.971 - 0.8$
$= 0.171$

Information Gain if we split using Age label is 0.171 when color is Purple

Label	Information Gain when Color = Purple
Size	0.0
Act	0.02
Age	0.171

From above values, we can witness that Age label has highest information gain than others when Color = Purple. Hence, we can proceed with Age as Root node after the Root Node Split (Color).



Sivaranjani Prabasankar A20436206

2). List at least two methods to alleviate overfitting in the decision tree learning? describe them in detail. [10]

Over-fitting occurs when the model is over-trained by the training set. It may show a high accuracy on training set, but bad performance on test set.

In decision trees, over-fitting occurs when the tree is modelled so perfectly as it fit all samples in the training data set. This effects the accuracy when predicting samples using test data which are not part of the training set.

There are 2 approaches to alleviate over-fitting in Decision Tree:

1) Stop earlier ➔ Stop growing the tree earlier

   Type 1: Limiting the level/depth of tree Early

   This stopping may underfit the model if we stop too early. Current split in the tree may be of little benefit, but subsequent splits more significantly reduce the error.

   Type 2: Check error at each split

   At each stage of splitting the tree, we check for error. If the error does not decrease significantly then we can stop building the tree.

2) Post-prune ➔ First allow over-fit and then re-structure the tree

   It is the process of trim off the branches of the tree i.e., removing the decision branches and subtrees are generated in a way that the overall accuracy is not disturbed.

   Step1: First build the tree completely with all complex branches.

   Step2: Then from tree's leaf node, if the accuracy without splitting is higher than the accuracy with splitting, replace the subtree with a leaf node and label it by the majority class.

   Then continue trimming the tree accordingly to optimize the classification accuracy.

2. Read the paper "Ensemble-based classifiers", and answer the following questions [50]

Sivaranjani Prabasankar A20436206

1). What is the ensemble method? And why or what are the motivations to develop ensemble methods? [10]

Ensemble is a Supervised Learning concept in which the idea is to train and integrate multiple models using the same learning algorithm. Here to solve a problem we are constructing multiple machine learning models are strategically.

Major Components

Training set, Base Inducer, Diversity Generator, Combiner

Types

1) **Sequential ensemble methods**
   - ⇒ The base learners are generated sequentially (e.g. AdaBoost).
   - ⇒ The overall performance can be boosted by weighing previously mislabeled examples with higher weight.

2) **Parallel ensemble methods**
   - ⇒ The base learners are generated in parallel (e.g. Random Forest).

3) **Homogeneous ensemble methods**

⇒ If ensemble methods use a single base learning algorithm to produce base learners of the same type, then it is called as homogeneous ensembles.

4) **Heterogenous ensemble methods**

⇒ If ensemble methods use a single base learning algorithm to produce base learners of the different type, then it is called as heterogeneous ensembles

Motivations

- The aim for ensemble method is to improve the prediction performances.
- This method focusses to weigh several individual classifiers and combine them in order to obtain a classifier that outperforms every one of them.
- It can also be used for improving the quality and robustness of clustering algorithms.
- The basic motivation of sequential methods is to exploit the dependence between the base learners.
- The basic motivation of parallel methods is to exploit independence between the base learners since the error can be reduced dramatically by averaging.
- The base learners must be as accurate as possible and as diverse as possible for ensemble methods to be more accurate than any of its individual members.

Sivaranjani Prabasankar A20436206

3) Introduce how AdaBoost and Bagging work. What are the differences between these two methods, e.g. purposes [10]?

AdaBoost (Adaptive Boosting) algorithm by showing that a strong classifier in the probably approximately correct (PAC) sense can be generated by combining "weak" classifiers (simple classifiers whose classification performance is only slightly better than random classification). It improves the simple boosting algorithm via an iterative process. It converts the weak learners into strong classifier and trains sequentially.

Methodology: The main idea behind this algorithm is to give more focus to patterns that are harder to classify. The amount of focus is quantified by a weight that is assigned to every pattern in the training set. Initially, the same weight is assigned to all the patterns. In each iteration the weights of all misclassified instances are increased while the weights of correctly classified instances are decreased. Therefore, the weak learner is forced to focus on the difficult instances of the training set by performing additional iterations and creating more classifiers. Furthermore, a weight is assigned to every individual classifier. This weight measures the overall accuracy of the classifier and is a function of the total weight of the correctly classified patterns. Thus, higher weights are given to more accurate classifiers. These weights are used for the classification of new patterns.

Bagging is a technique that improves the accuracy of a classifier by generating a composite model that combines multiple classifiers all of which are derived from the same inducer as boosting. Both methods follow a voting approach, which is implemented differently, in order to combine the outputs of the different classifiers. It creates an aggregated model with less variance.

Differences

1) **In bagging,** each instance is chosen with equal probability, while in boosting, instances are chosen with a probability that is proportional to their weight.
   In boosting, each classifier is influenced by the performance of those that were built prior to its construction. Specifically, the new classifier pays more attention to classification errors that were done by the previously built classifiers where the amount of attention is determined according to their performance.
2) **Bagging** requires an unstable learner as the base inducer.
   Boosting inducer instability in not required, only that the error rate of every classifier be kept below 0.5.
3) When the amount of noise is large, Boosting sometimes performs worse than Bagging. Moreover, Boosting outperforms Bagging when the noise level is small.

Sivaranjani Prabasankar A20436206

3) Compare decision trees and Random Forest, and discuss the advantages of random Forest [10]

Decision Trees

⇒ Decision Tree is a supervised, non- parametric machine learning algorithm.
⇒ Used for both classification as well as regression problems.
⇒ It is a graphical representation of tree like structure with all possible solutions.
⇒ It helps to reach a decision based on certain conditions.
⇒ Decision on how to split heavily impacts accuracy of decision tree.
⇒ It splits nodes based on available input variables. Selects the input variable resulting in best homogenous dataset.
⇒ Uses CART – Classification and Regression Tree which uses Gini Index (impurity measure) and Information Gain Index to build trees.

Random Forest

⇒ Random Forest increases predictive power of the algorithm and helps prevent overfitting.
⇒ It is the simplest and widely used algorithm.
⇒ Used for both classification and regression.
⇒ It is an ensemble of randomized decision trees.
⇒ Each decision tree gives a vote for the prediction of target variable. Random forest choses the prediction that gets the most vote.
⇒ In random forest we use multiple random decision trees for a better accuracy.

Random Forest is an ensemble bagging algorithm to achieve low prediction error. It reduces the variance of the individual decision trees by randomly selecting trees and then either average them or picking the class that gets the most vote.

Advantages of Random Forest

1. Works well with missing data still giving a better predictive accuracy
2. Efficient in handling a very large number of inputs and can be trusted with the results.
3. Prediction is based on input features considered important for classification.
4. Random forest, being a combination of many decision trees, provides reliable results.
5. Though Random forest was developed for decision trees, it can be applied to all the classifiers and it is faster compared to the traditional decision tree.
6. Random forest will reduce variance part of error rather than bias part, so on a given training data set decision tree may be more accurate than a random forest. But on an unexpected validation data set, Random forest always wins in terms of accuracy.

Sivaranjani Prabasankar A20436206

4). Introduce how to combine the results together in the ensemble methods [20]

There are two main methods for combining the base-classifiers' outputs.

1) WEIGHTING METHODS

Weighting methods are useful if the base-classifiers perform the same task and have comparable success.

When combining classifiers with weights, a classifier's classification has a strength proportional to its assigned weight. The assigned weight can be fixed or dynamically determined for the specific instance to be classified.

a) Majority Voting

⇒ A classification of an unlabeled instance is performed according to the class that obtains the highest number of votes (the most frequent vote).
⇒ This method is also known as the plurality vote (PV) or the basic ensemble method (BEM).
⇒ This approach is the most frequently used as a combining method for comparing newly proposed methods.

b) Performance weighting

⇒ The weight of each classifier can be set proportional to its accuracy performance on a validation set

c) Distribution summation

Here we are combining method is to sum up the conditional probability vector obtained from each classifier. The class will be selected according to the highest value in the total vector.

d) Bayesian combination

In Bayesian combination method the weight associated with each classifier is the posterior probability of the classifier given the training set.

e) Dempster–Shafer

The idea of using the Dempster–Shafer theory of evidence is to combine the classifiers using notion of basic probability assignment defined for a certain class at the given instance.

f) Vogging

This vogging (Variance Optimized Bagging) approach aims to optimize a linear combination of base-classifiers to aggressively reduce variance while attempting to preserve a prescribed accuracy.

Sivaranjani Prabasankar A20436206

g) Naïve Bayes

It makes the use of Naïve Bayes rule for combining various classifiers.

h) Entropy weighting

In this method we are having to assign each classifier a weight that is inversely proportional to the entropy of its classification vector.

i) Density-based weighting

If the various classifiers were trained using datasets obtained from different regions of the instance space and making it useful to weight the classifiers according to the probability of sampling by classifier.

j) DEA weighing method (Data Envelop Analysis)

The idea is to figure out the set of efficient classifiers. Here the weights should not be specified according to a single performance measure but should be based on several performance measure. As there is a trade-off among the various performance measures and the DEA is employed in order to figure out the set of efficient classifiers.

k) Logarithmic opinion pool

According to logarithmic opinion pool, the selection of the preferred class is performed according to the given formula.

$$Class(x) = \underset{c_i \in dom(y)}{\operatorname{argmax}} \sum_{k:c_i = \underset{c_j \in dom(y)}{\operatorname{argmax}} \hat{P}_{M_k}(y=c_j|x)} \hat{P}_{M_k}(x)$$

Where The estimation of ˆPMk (x) depends on the classifier representation and cannot always be estimated.

l) Gating network

Each expert outputs the conditional probability of the target attribute given the input instance. A gating network is responsible for combining the various experts by assigning a weight to each network. These weights are not constant but are functions of the input instance. The gating network selects one or a few classifiers which appear to have the most appropriate class distribution. In fact, each specializes on a small portion of the input space.

An extension to the basic mixture of experts, known as hierarchical mixtures of experts (HME), has been proposed which decomposes the space into sub-spaces, and then recursively decomposes each sub-space into sub-spaces. Variations of the basic mixtures of classifier methods have been developed to accommodate specific domain problems.

Sivaranjani Prabasankar A20436206

m) Order statistics

Order statistics can be used to combine classifiers. This offer the simplicity of a simple weighted combination method together with the generality of meta-combination methods. The robustness of this method is helpful when there are significant variations among classifiers.

2) META-LEARNING METHODS

Meta-learning methods are best suited for cases when certain classifiers consistently correctly classify, or consistently misclassify at certain instances.

a) Stacking

⇒ This method creates a meta-dataset containing a tuple for each tuple in the original dataset. Stacking is a technique for achieving the highest generalization accuracy.
⇒ It induces classifiers which are reliable, and which are not with the help of using a meta-learner.
⇒ Stacking is usually employed to combine models built by different inducers
⇒ It uses the predicted classifications by the classifiers as the input attributes instead of original input attribute.
⇒ The target attribute remains as in the original training set.
⇒ A test instance is first classified by each of the base classifiers, these classifications are then fed to a meta-level training set from which a meta-classifier is produced.
⇒ This classifier then combines the different predictions into a final one.

b) Arbiter trees

The idea of an arbiter is to provide an alternate classification when the base classifiers present diverse classifications. An arbiter tree is built in a bottom-up approach. Initially, the training set is randomly partitioned into k disjoint subsets. The arbiter is induced from a pair of classifiers and recursively a new arbiter is induced from the output of two arbiters.

⇒ The creation of the arbiter is performed as follows: for each pair of classifiers, the union of their training dataset is classified by the two classifiers.
⇒ A selection rule compares the classifications of the two classifiers and selects instances from the union set to form the training set for the arbiter.
⇒ The arbiter is induced from this set with the same learning algorithm used in the base level.
⇒ This arbiter together with an arbitration rule decides on a final classification outcome based upon the base predictions.

c) Combiner trees

The way combiner trees are generated is very similar to arbiter trees as both follow a bottom-up approach. However, in combiner, instead of an arbiter, is placed in each non-leaf node of

a combiner tree. In the combiner strategy, the classifications of the learned base classifiers form the basis of the meta-learner's training set. A composition rule determines the content of training examples from which a combiner or a meta-classifier will be generated.

d) Grading

This technique uses graded classifications as meta-level. The method transforms the classification made by the k different classifiers into k training sets by using the instances k times and attaching them to a new binary class in each occurrence. This class indicates whether the kth classifier yielded a correct or incorrect classification, compared to the real class of the instance. For every classifier, one meta-classifier is trained whose aim is to indicate when each classifier tends to misclassify a given instance. At classification time, each classifier tries to classify an unlabeled instance. The final classification is obtained by combining the outputs of the classifiers that are recognized as correct by the meta-classification schemes.