* Problem - 1:

1) For calculating total entropy, we know that we have 5 datapoints that have a negative class value while we have 4 datapoints that have a positive class value

Hence total entropy $E = -\frac{4}{9}\log_2(4/9) - \frac{5}{9}\log_2(5/9)$

$\therefore \boxed{E = 0.9911}$

Lets calculate information gain if we split using attribute $a_1$.

We know,

$$\text{gain} = E - \sum_i \frac{N_i}{m} E(i)$$

For $a_1$,

| $a_1$ \| C | + | - |
|---|---|---|
| T | 3 | 1 |
| F | 1 | 4 |

$$\therefore \text{gain} = E - \frac{4}{9}E(T) - \frac{5}{9}E(F)$$

$$E(T) = -\frac{3}{4}\log_2(3/4) - \frac{1}{4}\log_2(1/4) = 0.8112$$

$$E(F) = -\frac{1}{5}\log_2(1/5) - \frac{4}{5}\log_2(4/5) = 0.7219$$

$$\text{gain} = E - \frac{4}{9}(0.8112) - \frac{5}{9}(0.7219)$$

$$= 0.9911 - 0.3605 - 0.4011$$

$$= 0.2295$$

gain for splitting using attribute $a_1$ = $\text{gain}_{a_i}$ = 0.2295

$$\boxed{\text{gain}_{a_1} = 0.2295} \quad - (1)$$

Performing binary division at distinct values,

→ $a_2$ (1.0)

| 1.0 \| C | + | − |
|---|---|---|
| >= | 4 | 5 |
| < | 0 | 0 |

$$E(>=1) = -\frac{4}{9}\log_2(4/9) - \frac{5}{9}\log_2(5/9)$$

$$= 0.9911 \qquad E = 0.9911$$

$$\boxed{\text{gain}_{a_2=1.0} = 0}$$

→ $a_2$ (3.0)

| 2.0 \| C | + | − |
|---|---|---|
| >= | 3 | 5 |
| < | 1 | 0 |

$$E(>=3.0) = -\frac{3}{8}\left(\log_2(3/8)\right) - \frac{5}{8}\left(\log_2(5/8)\right)$$

$$= 0.9544$$

$$E(<3.0) = 0$$

$$\text{gain}_{a_2=3.0} = E - \sum_i \frac{m_i}{m} E(i)$$

$$= E - \frac{8}{9}(0.9544)$$

$$= 0.9911 - 0.8484$$

$$= 0.1427$$

$$\text{gain}_{a_2=3.0} = 0.1427$$

→ $A_2$ (e.g.)

| $A_2$ / C | ↑ | - |
|---|---|---|
| > 4 | 3 | 4 |
| < 4 | 1 | 1 |

Expected = E([3,1]) & E([1,1])

$$E(>4) = -\frac{3}{4}\log_2\left(\frac{3}{4}\right) - \frac{1}{4}\log_2\left(\frac{1}{4}\right)$$

$$= 0.9852$$

$$E(<4) = -\frac{1}{2}\log_2\left(\frac{1}{2}\right) - \frac{1}{2}\log_2\left(\frac{1}{2}\right)$$

$$= 1$$

$$\text{gain} = E - \sum_i \frac{m_i}{m} E(i)$$

$$= 0.9911 - \frac{4}{9}(0.9852) - \frac{2}{9}(1)$$

$$= 0.9911 - 0.2222 - 0.2463$$

$$= 0.0026$$

$$\text{gain}_{A_2 = 4.0} = 0.0026$$

→ $a_2$ (5.0)

| 5.0 \ C | + | − |
|---|---|---|
| >= | 2 | 4 |
| < | 2 | 1 |

$$E(>=5) = -\frac{2}{6}\log_2(2/6) - \frac{4}{6}\log_2(4/6)$$

$$E(<5) = -\frac{2}{3}\log_2(2/3) - \frac{1}{3}\log_2(1/3)$$

$$E(>=5) = 0.9183$$

$$E(<5) = 0.9183$$

$$\text{gain}_{a_2=5.0} = E - \frac{6}{9}(E(>=5)) - \frac{3}{9}(E<5)$$

$$= 0.9911 - \frac{2}{3}(0.9183) - \frac{1}{3}(0.9183)$$

$$= 0.0728$$

$$\text{gain}_{a_2=5.0} = 0.0728$$

→ $a_2$ (6.0)

| 6.0 \ C | + | − |
|---|---|---|
| >= | 2 | 2 |
| < | 2 | 3 |

$$E(>=6) = -\frac{2}{4}\log_2(2/4) - \frac{2}{4}\log_2(2/4) = 1$$

$$E(<6) = -\frac{2}{5}\log_2(2/5) - \frac{2}{5}\log_2(2/5) = 0.971$$

$$\text{gain}_{a_2 6.0} = E - E(>=6)\cdot\frac{4}{9} - \frac{5}{9}E(<6)$$

$$= E - \frac{4}{9}(1) - \frac{5}{9}(0.971)$$

$$= 0.9911 - 0.4444 - 0.5394$$

$$= 0.072$$

$$\boxed{\text{gain}_{a_2=6} = 0.072}$$

→ $a_2$ (7.0)

| 7.0 | C | f | - |
|---|---|---|---|
| >= | | 1 | 2 |
| < | | 3 | 3 |

$$E(>=7) = -\frac{1}{3}\log_2(1/3) - \frac{2}{3}\log_2(2/3)$$

$$= 0.9183$$

$$E(<7) = -\frac{1}{2}\log_2(1/2) - \frac{1}{2}\log_2(1/2)$$

$$= 1$$

$$\text{gain}_{a_2=7.0} = E - \frac{3}{9}(E(>=7)) - \frac{6}{9}E(<7)$$

$$= 0.9911 - \frac{1}{3}(0.9183) - \frac{2}{3}(1)$$

$$= 0.0183$$

$$\text{gain}_{a_2=7.0} = 0.0183$$

→ $a_2(8.0)$

| 8.0 \ C | + | − |
|---|---|---|
| >= | 0 | 1 |
| < | 4 | 4 |

$E(>=8) = 0$

$E(<8) = -\frac{1}{2}\log(1/2) - \frac{1}{2}\log(1/2)$

$= 1$

$gain_{a_2=8.0} = E - \frac{8}{9}(1)$

$= 0.1023$

| $gain_{a_2=8.0} = 0.1023$ |
|---|

gain for $a_1$ is greater than any of the gains for $a_2$. Hence we should first select $a_1$ for splitting for the decision tree.

→ Hence $a_1$ is the best attribute to start splitting.

2) Since instances are just used as serial numbers, we should not use it as an attribute for splitting. Suppose we use instance as an attribute. An attribute usually show something about the class distribution. If the attributes are jumbled then there would'nt be any change to the class distribution since they are linked to the class values. If we jumble the dataset, the values of instances don't change but the class distribution changes instantly. So it is not advisable to use instance as an attribute while splitting.

* Problem - 2 :

| | | Class Label | |
|---|---|---|---|
| A | B | + | - |
| T | T | 0 | 20 |
| T | F | 20 | 10 |
| F | T | 15 | 0 |
| F | F | 0 | 35 |

| Cost matrix | Attribute value | | |
|---|---|---|---|
| Actual class | | T | F |
| | + | -1 | 100 |
| | - | 0 | -10 |

1)

| A | B | + | - |
|---|---|---|---|
| T | T | 0 | 20 |
| T | F | 20 | 10 |
| F | T | 15 | 0 |
| F | F | 0 | 35 |

For A:

| A | + | - |
|---|---|---|
| T | 20 | 30 |
| F | 15 | 35 |

| | |
|---|---|
| + | 35 |
| - | 65 |

$$Gini_{orig} = 1 - \left(\frac{35}{100}\right)^2 - \left(\frac{65}{100}\right)^2$$

$$= 0.455$$

$$\boxed{Gini_{orig} = 0.455}$$

$$Gini_{split} = \sum_{i}^{n} \frac{n_i}{n} Gini(i)$$

$Gini(T)$

$$= \frac{50}{100}\left(1 - \left(\frac{20}{50}\right)^2 - \left(\frac{30}{50}\right)^2\right)$$

$Gini(F)$

$$+ \frac{50}{100}\left(1 - \left(\frac{35}{50}\right)^2 - \left(\frac{15}{50}\right)^2\right)$$

$$= \frac{1}{2}(0.48) + \frac{1}{2}(0.42) = 0.24 + 0.21$$

$$\boxed{Gini_{split} = 0.45}$$

For B:

| B / Class | + | − |
|---|---|---|
| T | 15 | 20 |
| F | 20 | 45 |

35
65

$$Gini_{org} = 1 - \left(\frac{35}{100}\right)^2 - \left(\frac{65}{100}\right)^2$$

$$= 0.455$$

$$Gini_{split} = \sum_{i}^{n} \frac{n_i}{n} Gini(i)$$

$$= \frac{35}{100}\left(1 - \left(\frac{15}{35}\right)^2 - \left(\frac{20}{35}\right)^2\right) + \frac{65}{100}\left(1 - \left(\frac{20}{65}\right)^2 - \left(\frac{45}{65}\right)^2\right)$$

($Gini(T)$ over the first bracket, $Gini(F)$ over the second)

$$= 0.35(0.4898) + 0.65(0.4260)$$

$$= 0.1714 + 0.2769$$

$$= 0.4483$$

$$\boxed{Gini_{split} = 0.4483}$$

→ Since Ginisplit for B is lesser than Ginisplit for A, we should choose B for splitting.

→ Hence splitting by attribute B, has to be the best way according to Gini Indices.

2)

For A:

| | + | − |
|---|---|---|
| T | 20 | 30 |
| F | 15 | 35 |

| CM | T | F |
|---|---|---|
| + | −1 | 100 |
| − | 0 | −10 |

from above tables,

cost of splitting A = 20(−1) + 30(0) + 100(15) + (35)(−10)

$Cost_{Asplit}$ = −20 + 1500 + (−350)

$Cost_{Asplit} = 1130$

P.T.O.

For B :

| B | + | − |
|---|---|---|
| T | 15 | 20 |
| F | 20 | 45 |

| CM | T | F |
|---|---|---|
| + | -1 | 100 |
| − | 0 | -10 |

Cost of splitting B,

$$\text{Cost}_{B\,\text{split}} = 15(-1) + 20(0) + 20(100) + (-10)(45)$$

$$= 2000 - 15 - 450$$

$$= 1535$$

Since cost of splitting A is lower, we will choose attribute A for splitting.

## Problem - 3

| ID | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| X | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1 |
| Y | 1 | 1 | 1 | -1 | -1 | -1 | -1 | -1 | 1 | 1 |

H1 : if $X \leq 0.35 \rightarrow Y = 1$, else $Y = -1$

H2 : if $X \leq 0.75 \rightarrow Y = -1$, else $Y = 1$

H3 : if $X \leq 0.3$ or $X > 0.95 \rightarrow Y = 1$ else $Y = -1$

7

a)

| X | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|
| y | 1 | 1 | 1 | -1 | -1 | -1 | -1 | -1 | 1 | 1 |
| $h_1(x)$ | 1 | 1 | 1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |

We are going to initialize weights.

| X | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|
| $D_t(i)$ | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 |

always, $D_t(i) = \frac{1}{n}$ (for first step)

incorrect predictions := 0.9 and 1 (X)

hence no. of incorrect predictions = 2

error = $\sum D_t(i) \cdot \delta(h_1(x) \neq y)$

= 0.1 + 0.1 = 0.2

$$\boxed{\varepsilon = 0.2}$$

$$\alpha = \frac{1}{2} \ln\left(\frac{1-\varepsilon}{\varepsilon}\right) = \frac{1}{2} \ln 4$$

= 0.6931

$$\boxed{\alpha = 0.6931}$$

correct classification = $D_2(i) = D_1(i) \exp(-\alpha y_i h_1(x_i))$
(new weights)

$$D_2(i) = 0.1 \times e^{(-0.6931)}$$

correct classification

$$= 0.05$$

incorrect classification = $D_2(i) = D_1(i) \exp(-\alpha y h(x))$

$$D_2(i) = 0.1 \times e^{(0.6931)}$$

incorrect classification = 0.2

∴ new weights

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| X | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1.0 |
| Y | 1 | 1 | 1 | -1 | -1 | -1 | -1 | -1 | 1 | 1 |
| h(x) | 1 | 1 | 1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| $D_t$ | 0.05 | 0.05 | 0.05 | 0.05 | 0.05 | 0.05 | 0.05 | 0.05 | 0.2 | 0.2 |

P.T.O.

(b) For hypothesis $H_2$

| X | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1.0 |
|---|---|---|---|---|---|---|---|---|---|---|
| Y | 1 | 1 | 1 | -1 | -1 | -1 | -1 | -1 | 1 | 1 |

Assigning equal weights.

| X | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1.0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $D_1$ | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 |

Values of Y according to $H_2$, ie $h_2(x)$

| X | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|
| $h_2(x)$ | -1 | -1 | -1 | -1 | -1 | -1 | -1 | 1 | 1 | 1 |
| Y | 1 | 1 | 1 | -1 | -1 | -1 | -1 | -1 | 1 | 1 |

instances where $h_2(x)$ predicts incorrectly = 4 (0.1, 0.2, 0.3, 0.8)

$\therefore$ error $= \sum_i D_i \, \delta(h(x) \neq y)$

$= 0.1 + 0.1 + 0.1 + 0.1 + 0.1$

$= 0.5$

$\boxed{\varepsilon = 0.4}$

$\alpha = \frac{1}{2} \ln\left(\frac{1-\varepsilon}{\varepsilon}\right)$

$= \frac{1}{2} \ln\left(\frac{0.6}{0.4}\right) = \frac{1}{2} \ln(3/2)$

$= \frac{1}{2}(.4054) = 0.2027$

$\boxed{\alpha = 0.2027}$

New weights = $D_{t+1}(i) = D_t(i) \exp(-\alpha y h(x))$

correct prediction = $0.1 \times e^{-(0.2027)}$

$= 0.0816$

incorrect prediction = $0.1 \times e^{(0.2027)}$

$= 0.1225$

| New weights | X | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1.0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Y | 1 | 1 | 1 | -1 | -1 | -1 | -1 | -1 | 1 | 1 |
| | h(x) | -1 | -1 | -1 | -1 | -1 | -1 | -1 | 1 | 1 | 1 |
| | $D_2$ | 0.1225 | 0.1225 | 0.1225 | 0.08 | 0.08 | 0.08 | 0.08 | 0.1225 | 0.08 | 0.08 |

(c) For hypothesis $h_3$

| X | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1.0 |
|---|---|---|---|---|---|---|---|---|---|---|
| Y | 1 | 1 | 1 | -1 | -1 | -1 | -1 | -1 | 1 | 1 |

Initializing weights

| X | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|
| $D_t$ | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 |

Predicting y using $h_3$

| X | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1.0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $h_3$ | 1 | 1 | 1 | -1 | -1 | -1 | -1 | -1 | -1 | 1 |
| Y | 1 | 1 | 1 | -1 | -1 | -1 | -1 | -1 | 1 | 1 |

number of instances with incorrect prediction = 1 (0.9)

∴ error = $\sum_i D_t(i)\, \delta(h(x) \neq y)$

= 0.1

$$\boxed{\varepsilon = 0.1}$$

$$\alpha = \frac{1}{2} \ln\left(\frac{1-\varepsilon}{\varepsilon}\right) = \frac{1}{2} \ln(0.9/0.1)$$

$$= \frac{1}{2} \ln(9) = (2.1972)/2 = 1.0986$$

$$\boxed{\alpha = 1.0986}$$

New weights, = $D_t(i) \exp(-\alpha y h(x))$

correct prediction = $0.1 \times e^{(-1.0986)}$

= 0.0333

incorrect prediction = $0.1 \times e^{(1.0986)}$

= 0.3

New weights:

| X | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 | 1.0 |
|---|---|---|---|---|---|---|---|---|---|---|
| Y | 1 | 1 | 1 | -1 | -1 | -1 | -1 | -1 | 1 | 1 |
| h(x) | 1 | 1 | 1 | -1 | -1 | -1 | -1 | -1 | -1 | 1 |
| $D_2$ | 0.033 | 0.033 | 0.033 | 0.033 | 0.033 | 0.033 | 0.033 | 0.033 | 0.3 | 0.03 |

2) To be precise, for all the data instances where $\gamma = 0.9$ and $\gamma = 1$ are reweighted, for the the data instances where $\gamma = 0.1, 0.2, 0.3$ and $0.8$ are reweighted and for $1/3$ the data instance where $\gamma = 0.9$ is only reweighted. Weights of all the data instances are changed but the others are changed for balancing the effect of these particular data instances which were incorrect.

* Problem - 4

Part I :

datapoints : { (1, 1, -), (2, 2, -), (1, 3, -), (4, 1, -), (4, 4, -), (5, 1, -), (3, 6, -), (8, 1, -), (9, 1, -), (6, 5, +), (2, 8, +), (7, 3, +), (9, 4, +), (9, 6, +), (8, 7, +), (7, 7, +), (6, 8, +), (5, 9, +), (8, 9, +) }

Let's suppose one test point belongs to class p which can be + or -.

datapoint to test $p_* = (5, 4, p)$

Calculating distance of pt from different datapoints :

$d((1,1,-)) = \sqrt{(5-1)^2 + (4-1)^2}$

$= \sqrt{25}$

$= 5$ $\therefore f((1,1,-)) = (5,-)$

$d((2,2,-)) = \sqrt{13}$

$= 3.605$ $\therefore f((2,2,-)) = (3.6,-)$

$d((1,3,-)) = \sqrt{17}$

$= 4.123$ $\therefore f((1,3,-)) = (4.123,-)$

$d((4,1,-)) = \sqrt{10}$

$= 3.16$ $\therefore f((4,1,-)) = (3.16,-)$

$d((4,4,-)) = \sqrt{1}$

$= 1$ $\therefore f((4,4,-)) = (1,-)$

$d((5,1,-)) = \sqrt{9}$

$= 3$ $\therefore f((5,1,-)) = (3,-)$

$d((2,6,-)) = \sqrt{13}$

$= 3.605$ $\therefore f((2,6,-)) = (3.6,-)$

$d((8,1,-)) = \sqrt{18}$

$= 4.24$ $\therefore f((8,1,-)) = (4.24,-)$

$d((9,1,-)) = \sqrt{25}$

$= 5$ $\therefore f((9,1,-)) = (5,-)$

$d((6,5,+)) = \sqrt{2}$

$= 1.41$ $\therefore f((6,5,+)) = (1.41, +)$

$d((2,8,+)) = \sqrt{25}$

$= 5.0$ $f((2,8,+)) = (5.0, +)$

$d((7,3,+)) = \sqrt{5}$

$= 2.23$ $f((7,3,+)) = (2.23, +$

$d((9,4,+)) = \sqrt{16}$

$= 4$ $f((9,4,+)) = (4, +)$

$d((9,6,+)) = \sqrt{20}$

$= 4.47$ $f((9,6,+)) = (4.47, +)$

$d((8,7,+)) = \sqrt{18}$

$= 4.24$ $f((8,7,+)) = (4.24, +)$

$d((7,7,+)) = \sqrt{13}$

$= 3.60$ $f((7,7,+)) = (3.6, +)$

$d((6,8,+)) = \sqrt{17}$

$= 4.123$ $f((6,8,+)) = (4.123, +$

$d((5,9,+)) = \sqrt{25}$

$= 5$ $f((5,9,+)) = (5, +)$

$d((8,9,+)) = \sqrt{34}$

$= 5.83$ $f((8,9,+)) = (5.83, +$

| point | class | distance |
|---|---|---|
| (1, 1) | - | 5 |
| (2, 2) | - | 3.6 |
| (1, 3) | - | 4.12 |
| (4, 1) | - | 3.16 ← |
| (4, 4) | - | 1.0 ← |
| (5, 1) | - | 3.0 ← |
| (5, 6) | - | 3.6 |
| (8, 1) | - | 4.24 |
| (9, 1) | - | 5 |
| (6, 5) | + | 1.41 ← |
| (5, 8) | + | 5 |
| (7, 3) | + | 2.23 ← |
| (9, 4) | + | 4 |
| (9, 6) | + | 4.47 |
| (8, 7) | + | 4.24 |
| (7, 7) | + | 3.6 |
| (6, 8) | + | 4.12 |
| (5, 9) | + | 5 |
| (8, 9) | + | 5.83 |

5 nearest :

(4, 4, -), (6, 5, +), (7, 3, +), (4, 4, -), (4, 1, -)

majority of the classes are negative from the 5- nearest neighbours, so the prediction is class '-'.

P. T. O.

2) 

| points | Manhattan distance (d) |
|---|---|
| (1, 1, '-') | 7 |
| (2, 2, '-') | 5 |
| (1, 3, '-') | 5 |
| (4, 1, '+') | 4 |
| (4, 4, '-') | 1 → min |
| (5, 1, '-') | 3 → min |
| (2, 6, '-') | 5 |
| (8, 1, '-') | 6 |
| (9, 1, '-') | 7 |
| (6, 5, '+') | 2 → min |
| (7, 8, '+') | 7 |
| (7, 3, '+') | 3 → min |
| (9, 4, '+') | 4 |
| (9, 6, '+') | 6 |
| (8, 7, '+') | 6 |
| (7, 7, '+') | 5 |
| (6, 8, '+') | 5 |
| (5, 9, '+') | 5 |
| (8, 9, '+') | 8 |

| (4, 4, '-') | (5, 1, '-') | (6, 5, '+') | (7, 3, + |
|---|---|---|---|
| ↓ | ↓ | ↓ | ↓ |
| - | - | + | + |
| ↓ | ↓ | ↓ | ↓ |
| $w = \frac{1}{d^2} = \frac{1}{1} = 1$ | $w = \frac{1}{d^2} = \frac{1}{9}$ | $w = \frac{1}{d^2} = \frac{1}{4}$ | $w = \frac{1}{9}$ |

Since the weights are [1, 2, 3, 3] we consider both possibilities for 3.

Suppose we pick the 3 which has a '+' class, then the weights.

| | − | + |
|---|---|---|
| | 1 | 1/4 |
| | | 1/9 |
| T : | 1 | 13/36 |

$w(-) > w(+)$

Hence prediction would be −.

Suppose we pick the 3 which has a '−' class, then the weights

| | − | + |
|---|---|---|
| | 1 | 1/4 |
| | 1/9 | |
| T : | 10/9 | 1/4 |

$w(-) > w(+)$

Hence prediction would be −.

→ Hence in both cases prediction is −.

# Problem 4. K-nearest Neighbor Classification

# Part II.

## Preparing the dataset

### Importing Libraries

```
import pandas as pd
```

### Importing the training sample

```
train_sample = pd.read_csv('train.csv')
```

```
train_values = train_sample.values
tn_X, tn_y = train_values[:, :-1], train_values[:, -1]
```

### Importing the testing sample

```
test_sample = pd.read_csv('test.csv')
```

### Removing unwanted columns from the testing sample

```
final_cols = ['x','y','z','actual-class',]
test_sample = test_sample[final_cols]
```

```
test_values = test_sample.values
tt_X, tt_y = test_values[:, :-1], test_values[:, -1]
```

---

## K-Nearest Neighbors Algorithm

---

# Problem 4_Part II

## (1)

### For k = 3:

```
from sklearn.neighbors import KNeighborsClassifier #Importing KNN and CM


#preparing a model
model = KNeighborsClassifier(n_neighbors = 3)

#training the model on the train sample
model.fit(tn_X, tn_y)

#predicting using the model
prediction = model.predict(tt_X)
print(prediction)
print(tt_y)
```

```
[1. 0. 0. 1. 0. 1. 0. 0. 0. 0. 1. 0. 0. 0. 1. 1. 1. 0. 0. 0.]
[1. 0. 0. 1. 0. 1. 0. 0. 0. 0. 1. 0. 0. 0. 0. 1. 1. 0. 0. 0.]
```

### Probability of the final prediction

```
probab = model.predict_proba(tt_X)
print(probab)
```

```
[[0.         1.        ]
 [0.66666667 0.33333333]
 [1.         0.        ]
 [0.         1.        ]
 [1.         0.        ]
 [0.         1.        ]
 [1.         0.        ]
 [1.         0.        ]
 [1.         0.        ]
 [1.         0.        ]
 [0.         1.        ]
 [1.         0.        ]
 [1.         0.        ]
 [1.         0.        ]
 [0.33333333 0.66666667]
 [0.         1.        ]
 [0.         1.        ]
 [1.         0.        ]
 [1.         0.        ]
 [1.         0.        ]]
```

## Confusion Matrix, accuracy, precision and F-measure

```
tp = 0
tn = 0
fp = 0
fn = 0

for x in range(len(prediction)):
  if(prediction[x] == 0 and tt_y[x] == 0):
    tn += 1
  if(prediction[x] == 1 and tt_y[x] == 1):
    tp += 1
  if(prediction[x] == 0 and tt_y[x] == 1):
    fn += 1
  if(prediction[x] == 1 and tt_y[x] == 0):
    fp += 1

total = tn + fn + tp + fp
cm = [[tp,fp],[fn,tn]]
accuracy = (tp + tn)/total
precision = tp/(tp + fp)
recall = tp/(tp + fn)
f_score = 2*precision*recall/(precision + recall)

print("Accuracy:",accuracy)
print("Precision:",precision)
print("Recall:",recall)
print("F-Measure:",f_score)
```

```
Accuracy: 0.95
Precision: 0.8571428571428571
Recall: 1.0
F-Measure: 0.923076923076923
```

## Problem 4_Part II

## (2)

```
#Implenting weighted KNN
w_knn = KNeighborsClassifier(n_neighbors=3, weights='distance', p=2, metric='euclidean')


# Training the KNN model
w_knn.fit(tn_X, tn_y)
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='euclidean',
                     metric_params=None, n_jobs=None, n_neighbors=3, p=2,
```

```
                                weights='distance')

#Predicting the Output
w_prediction = w_knn.predict(tt_X)
print(w_prediction)

     [1. 0. 0. 1. 0. 1. 0. 0. 0. 0. 1. 0. 0. 0. 1. 1. 1. 0. 0. 0.]


tp = 0
tn = 0
fp = 0
fn = 0

for x in range(len(w_prediction)):
  if(w_prediction[x] == 0 and tt_y[x] == 0):
    tn += 1
  if(w_prediction[x] == 1 and tt_y[x] == 1):
    tp += 1
  if(w_prediction[x] == 0 and tt_y[x] == 1):
    fn += 1
  if(w_prediction[x] == 1 and tt_y[x] == 0):
    fp += 1

total = tn + fn + tp + fp
cm = [[tp,fp],[fn,tn]]
accuracy = (tp + tn)/total
precision = tp/(tp + fp)
recall = tp/(tp + fn)
f_score = 2*precision*recall/(precision + recall)

print("Accuracy:",accuracy)
print("Precision:",precision)
print("Recall:",recall)
print("F-Measure:",f_score)

     Accuracy: 0.95
     Precision: 0.8571428571428571
     Recall: 1.0
     F-Measure: 0.923076923076923
```

Since the dataset extremely small, both of the models perform same. If dataset was huge maybe one of them would have been better, and my guess would have been the weighted KNN would've been more accurate

✓ 0s completed at 5:11 PM