CSCE 633 – Machine Learning

Homework 02

Amiya Ranjan Panda

UIN – 727006179

Q3. IMPLEMENTATION OF SUPPORT VECTOR MACHINE.

The dataset was in its raw form divided into features which takes values from {-1,1} and features which takes value from {-1,0,1}. So, the first step adopted is to implement one-hot encoding for the dataset. This is done as mentioned. If a feature fi have value {−1, 0, 1}, we create three new features fi,−1, fi,0, and fi,1. Only one of them can have value 1 and fi,x = 1 if and only if fi = x. For example, we transform the original feature with value 1 into [0, 0, 1]. In the given dataset, the features 2, 7, 8, 14, 15, 16, 26, 29 (index starting from 1) take three different values {−1, 0, 1}. Thereafter, in order to bring uniformity to the dataset, the rest of the features were transformed to {0,1} from {-1,1}. Then, the columns (2, 7, 8, 14, 15, 16, 26, 29) are deleted from the data frame. This ultimately gives us the processed dataset upon which we will perform machine learning operation. For this, I have randomly separated the data into train (2/3) and test (1/3) set.

Then, I experimented with different values of misclassification cost C, applying 3-fold cross validation on the train set for “linear”, “polynomial” , and “rbf” kernels and the results are mentioned below.

C = [2^-10, 2^-9, 2^-8 ,…….,2^9]

LINEAR KERNEL:

For cross-validation **(accuracy, time) per C** is given by

[(0.9076549210206561, 4.118379354476929),

(0.9023896314297286, 3.026970148086548),

(0.9068448764682058, 2.3912651538848877),

(0.9206156338598623, 2.0652191638946533),

(0.9258809234507898, 1.877018928527832),

(0.928311057108141, 1.678412914276123),

(0.9307411907654921, 1.5605700016021729),

(0.9319562575941677, 1.4896621704101562),

(0.9335763466990684, 1.3853704929351807),

(0.9339813689752936, 1.4173738956451416),

(0.9335763466990684, 1.5218169689178467),

(0.9339813689752936, 1.8282573223114014),

(0.934791413527744, 2.192711114883423),

(0.9351964358039693, 2.9317445755004883),

(0.9360064803564196, 4.205877780914307),

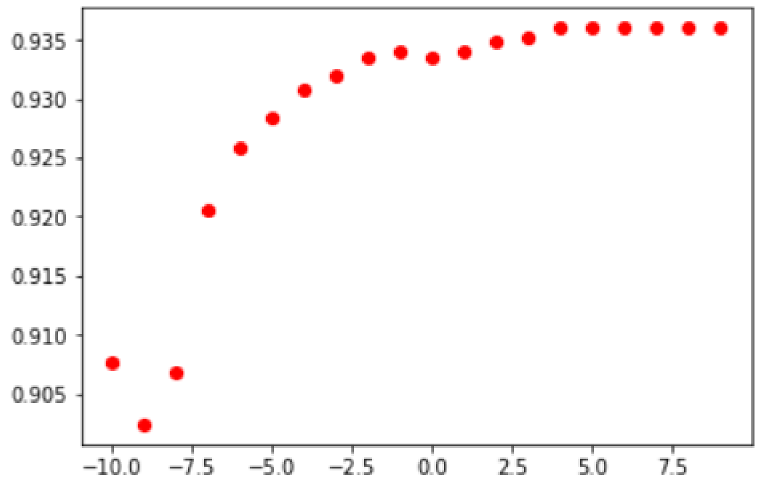
(0.9360064803564196, 6.522589206695557),

(0.9360064803564196, 10.026994466781616),

(0.9360064803564196, 17.090081930160522),

(0.9360064803564196, 30.599862337112427),

(0.9360064803564196, 59.58180546760559)]



As we can observe that for c = 32, we are getting maximum accuracy with less time. So, taking the same value and training again and testing with the **test set, we get the accuracy 0.930921052631579**.

POLYNOMIAL KERNEL:

For cross-validation **(accuracy, time) per C** is given by

[(0.5637910085054678, 10.316271305084229),

(0.5637910085054678, 10.271387815475464),

(0.5828270554880518, 10.207841634750366),

(0.9003645200486027, 9.821514129638672),

(0.91575536654516, 7.636522531509399),

(0.9088699878493317, 5.748750448226929),

(0.905224787363305, 4.385406494140625),

(0.9092750101255569, 3.4426074028015137),

(0.9210206561360875, 2.9219486713409424),

(0.9311462130417173, 2.5424270629882812),

(0.9339813689752936, 2.2714266777038574),

(0.9408667476711219, 2.108035087585449),

(0.9424868367760226, 1.965653896331787),

(0.9449169704333739, 1.8945419788360596),

(0.9501822600243013, 1.8594043254852295),

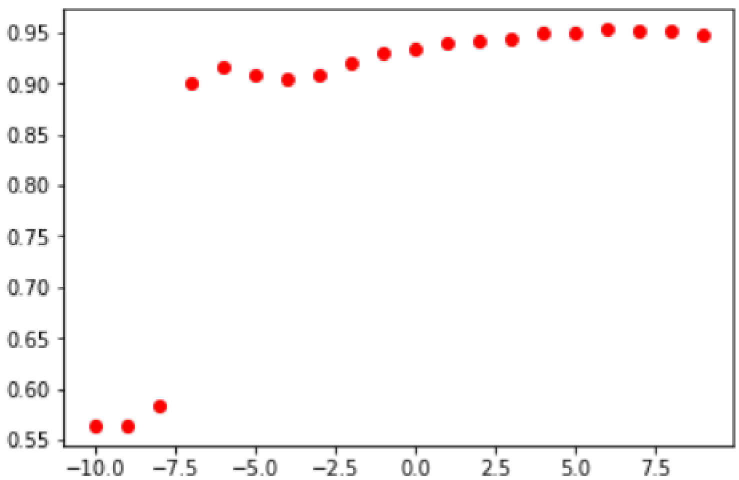
(0.9505872823005266, 1.8598318099975586),

(0.9534224382341029, 1.9636926651000977),

(0.9509923045767518, 2.1155261993408203),

(0.9509923045767518, 2.3274648189544678),

(0.9489671931956257, 2.5880088806152344)]



As we can observe that for c = 64, we are getting maximum accuracy with less time. So, taking the same value and training again and testing with the **test set, we get the accuracy 0.9629934210526315**.

RBF KERNEL:

For cross-validation **(accuracy, time) per C** is given by

[(0.5637910085054678, 7.8440024852752686),

(0.5637910085054678, 7.789108514785767),

(0.5637910085054678, 7.768023490905762),

(0.5637910085054678, 7.7883055210113525),

(0.680032401782098, 7.667978525161743),

(0.8833535844471446, 7.048547029495239),

(0.9044147428108545, 5.715918779373169),

(0.913730255164034, 4.7616355419158936),

(0.9189955447549615, 3.864760160446167),

(0.9214256784123127, 2.819880247116089),

(0.9299311462130417, 2.319140672683716),

(0.9360064803564196, 2.019554853439331),

(0.9392466585662211, 1.80995512008667),

(0.9437019036046983, 1.6348202228546143),

(0.947347104090725, 1.5201125144958496),

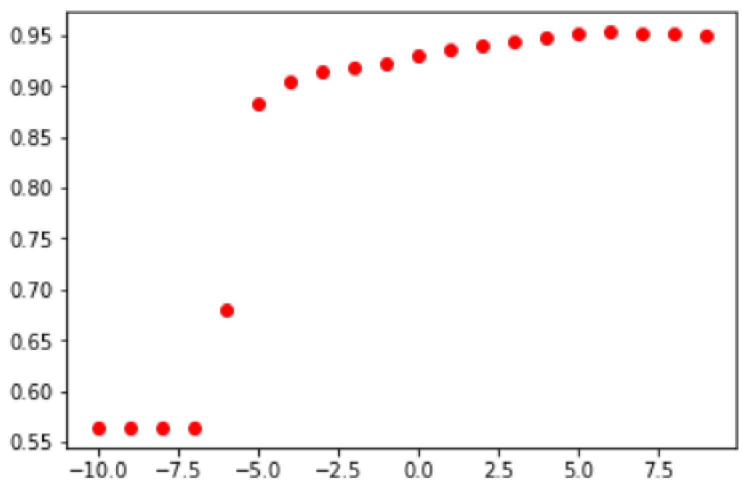
(0.9509923045767518, 1.45566725730896),

(0.9534224382341029, 1.4666175842285156),

(0.9509923045767518, 1.5008141994476318),

(0.9509923045767518, 1.635788917541504),

(0.9505872823005266, 1.784893274307251)]



As we can observe that for c = 64, we are getting maximum accuracy with less time. So, taking the same value and training again and testing with the **test set, we get the accuracy 0.9627192982456141**.

For “polynomial” and “rbf” kernels, **gamma value is kept in” auto” mode**. That is cross-validation is performed only across different values of C. We have observed that time increases dramatically after a certain value of C in all of the three kernels. And also, the accuracy decreases after a certain value of C for “polynomial” and “rbf” kernels. Whereas, for “linear” kernel, it becomes constant.

Q2. Decision Tree Implementation:

The task was to implement a basic version decision tree based on two metrices (for information gain) which are **1. Gini index 2. entropy** based. The code comprises of modules some of which takes are common to both and some take input as the type of matric and build tree accordingly.

First, we observer the type of data we have. The data represents the Wisconsin-Madison Breast cancer dataset which labels each row as a **Benign (2) or a Malignant (4)** one.

The count on the **original dataset for (Benign, Malignant) = [444, 239].**

After splitting the dataset into test and train samples,

For **train sample the division (Benign, Malignant) = [271, 187]**

For **test sample the division (Benign, Malignant) = [124, 101]**

The proportions in the test and the train samples are comparable enough to give a decent output to the model.

For the implementation part, I have used binary classification per node. That is, I have calculated the information gain for each value in the feature (1 to 10) by calculating separately for each row and returning the one with the highest information gain. This increases computation to an extent but gives a more accurate output.

The implementation includes pre-pruning using a lower threshold on the values of the splitting criterion for each branch. (**BONUS**)

For each matric (Gini index and entropy), accuracies are calculated for different stages for tree formation. This is achieved by making trees of different depths (here up to 20), then predicting and calculating the accuracy. Below given are the **accuracy (ON TEST DATA)** for both trees.

1. **Gini Index based:**

Results in next submission.

No1 (a.)

Temprature	Humidity	Sky	#Rain
Hot	High	Cloudy	9/10
Hot	High	Clear	5/10
Hot	Low	Cloudy	6/10
Hot	Low	Clear	3/10
Cool	High	Cloudy	7/10
Cool	High	Clear	2/10
Cool	Low	Cloudy	3/10
Cool	Low	Clear	1/10

for binary Classification, we need to take an assumption.

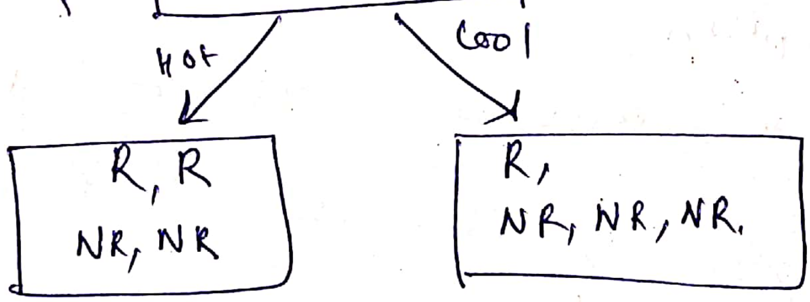
So, we have assumed that if,
#Rain < 6/10, it is "no Rain" (NR),
else it is "Rain" (R)

So, the updated table is as follows:→

Temprature	Humidity	Sky	# Rain
Hot	High	Cloudy	R
Hot	High	Clear	NR
Hot	Low	Cloudy	R
Hot	Low	Clear	NR
Cool	High	Cloudy	R
Cool	High	Clear	NR
Cool	low	Cloudy	NR
Cool	low	Clear.	NR

for Root :→

Considering $\boxed{\text{Temprature}}$ as the root node.



entropy (Hot) $= -\sum_i P_i \log P_i$

$$= -\left(\frac{2}{2+2}\times\log\left(\frac{2}{2+2}\right) + \frac{2}{2+2}\times\log\left(\frac{2}{2+2}\right)\right)$$

$$= -\left(\tfrac{1}{2}\log \tfrac{1}{2} + \tfrac{1}{2}\log\tfrac{1}{2}\right)$$

$\Rightarrow$ entropy (Hot) $=$ 1.

Similarly,

$$\text{entropy (Cool)} = -\left(\frac{1}{1+3}\times\log\left(\frac{1}{1+3}\right) + \frac{3}{1+3}\times\log\left(\frac{3}{4}\right)\right)$$

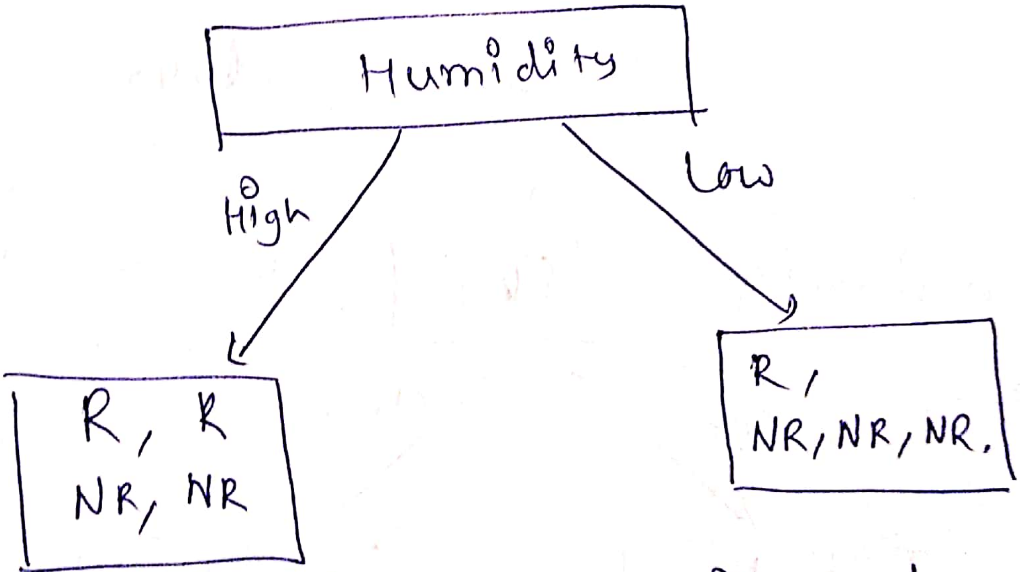
$$= -\left(\frac{1}{4}\log\frac{1}{4} + \frac{3}{4}\log\frac{3}{4}\right)$$

$$= 2 - \frac{3}{4}\times\log 3.$$

x

---

* Considering "Humidity" as root node.



As, the distribution of High branch is same as that of previously calculated "Hot",

entropy (High) $=$ 1.

Similarly,

$$\text{entropy (low)} = \text{entropy (Cool)} = 2 - \frac{3}{4}\times\log 3.$$

$H(\text{tempreture}) = \frac{4}{4+4} \times 1 + \frac{4}{4+4} \times \left(2 - \frac{3}{4}\log 3\right)$

(weighted avarage)

$= 1/2 + 1/2 \times (2 - 3/4 \log 3)$

$= 1/2 + 1 - 3/8 \times \log 3$

$= 3/2 - \frac{3}{8} \times \log 3.$

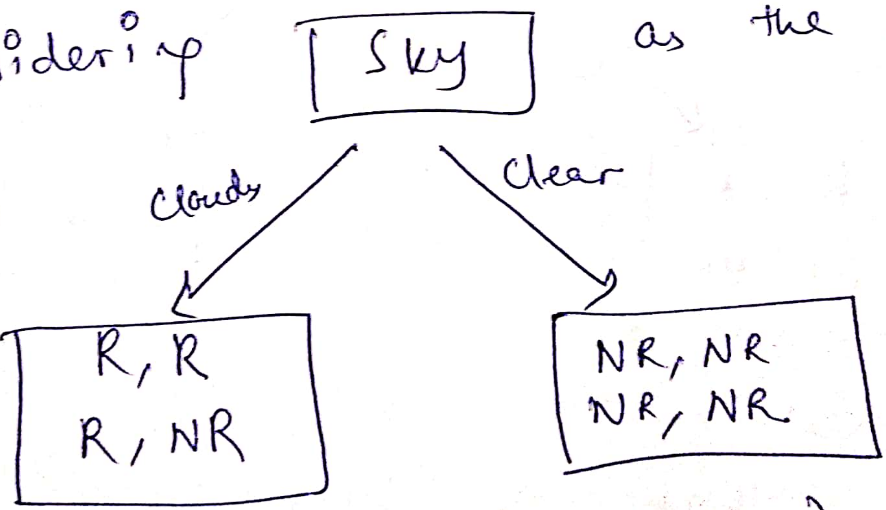
So,

$H(\text{Humidity}) = H(\text{tempreture})$

$= \frac{3}{2} - \frac{3}{8} \times \log 3.$

X

Now, Considering Sky as the splitting criteria



$\text{entropy}(\text{Cloudy}) = -\left(\frac{1}{1+3} \times \log\left(\frac{1}{1+3}\right) + \frac{3}{1+3} \times \log \frac{3}{1+3}\right)$

$= -\left(1/4 \log 1/4 + 3/4 \log 3/4\right)$

$= 2 - \frac{3}{4} \log 3.$

$$\text{entropy(clear)} = \left(\frac{0}{4} \log \frac{0}{4} + \frac{4}{4} \log \frac{4}{4} \right)$$

$$= 0.$$

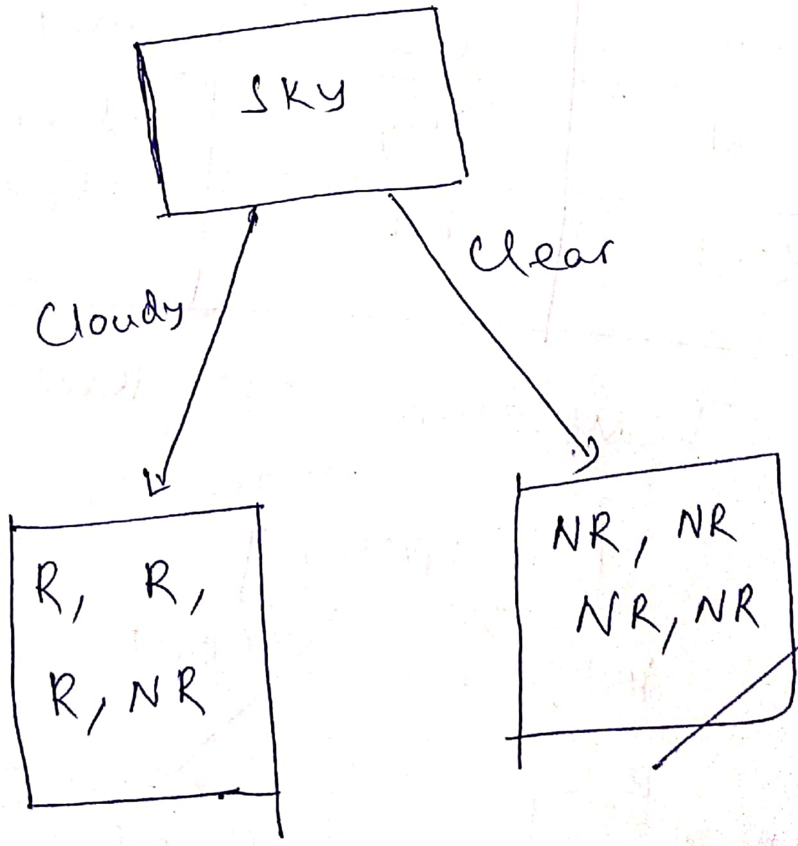
$$\Rightarrow H(\text{Sky}) = \frac{4}{4+4} \times \left(2 - \frac{3}{4} \log 3 \right) + \frac{4}{4+4} \times 0$$

$$= \frac{1}{2} \times \left(2 - \frac{3}{4} \log 3 \right).$$

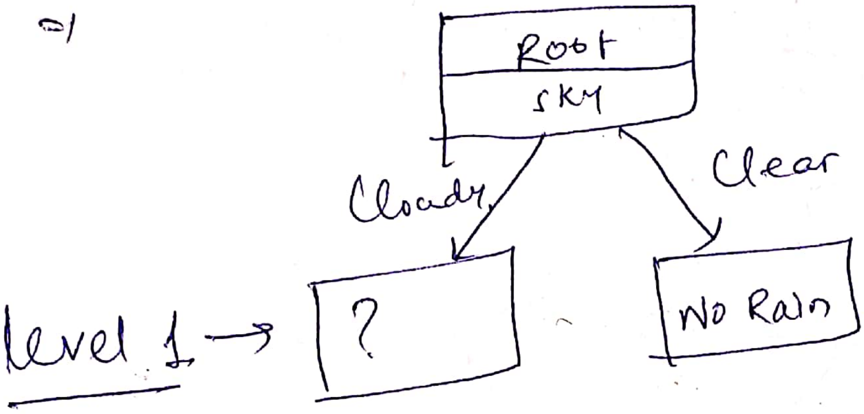
$$= 1 - 3/8 \log 3$$

By, Comparing H(Tempreture), H(Humidity) and H(Sky), we know that H(Sky) is the least. So, the root node is "SKY"

Now,



minimal node or leaf node with prediction "No Rain".

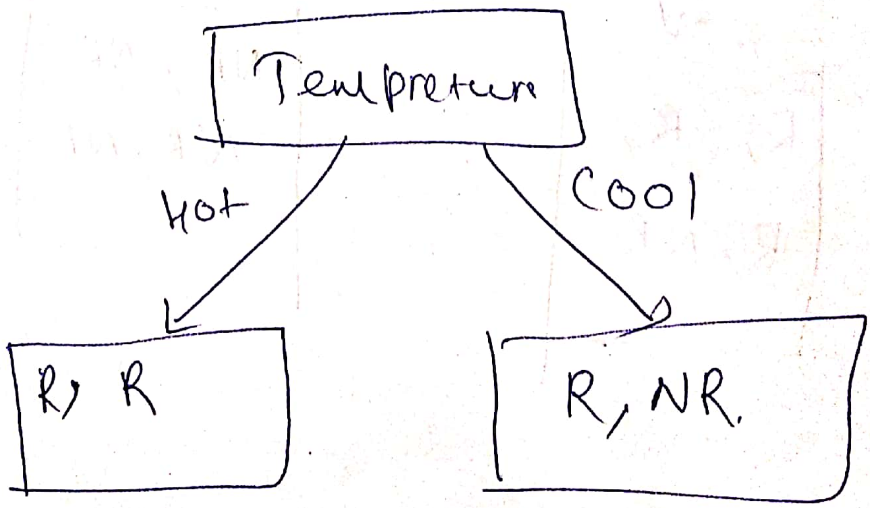


For Level 1

Updated table is,

Tempreture	Humidity	Sky	Rain
Hot	High	Cloudy	Rain
Hot	Low	Cloudy	Rain
Cool	High	Cloudy	Rain
Cool	Low	Cloudy	No Rain

Considering Tempreture as level 1 node,



$$\text{entropy}(\text{Hot}) = -\left(\frac{0}{2}\log\frac{0}{2} + \frac{2}{2}\log\frac{2}{2}\right)$$

$$= 0.$$

$$\text{entropy}(\text{Cool}) = -\left(\frac{1}{2}\log\frac{1}{2} + \frac{1}{2}\log\frac{1}{2}\right)$$

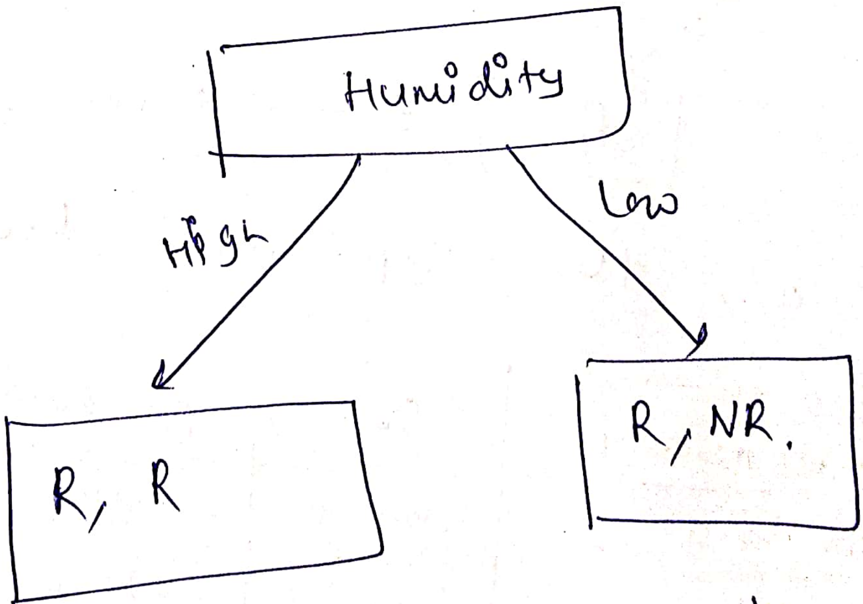
$$= 1.$$

$$H(\text{Tempreture}) = \frac{2}{2+2} \times 0 + \frac{2}{2+2} \times 1$$

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

$$= \frac{1}{2}.$$

Considering, Humidity as the level 1 node

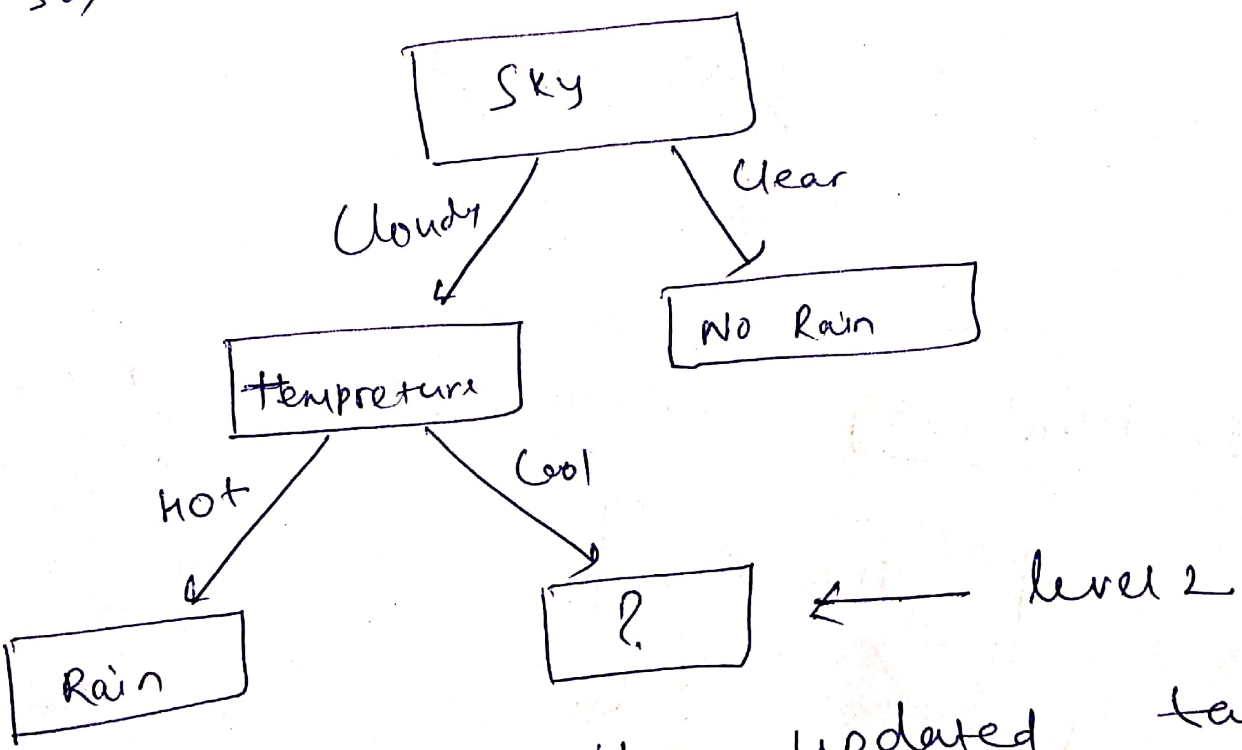


As, the distribution of Humidity is same as that of Tempreture, we can say that $H(\text{Humidity}) = 1/2$.

To disambiguate the situation created due to equal entropy, we will have to select randomly any "spliting criteria".

So, let us choose Temprature as the splitting criteria for level 1.

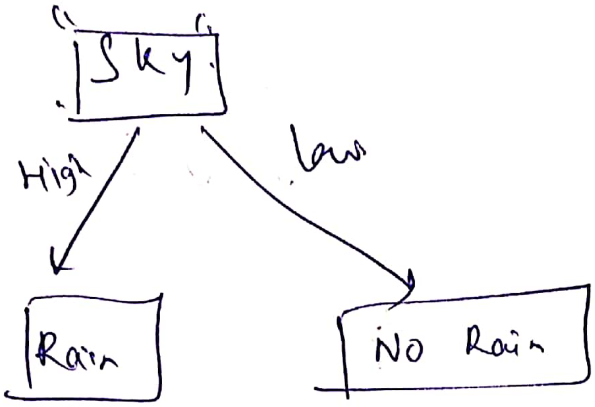
So, the updated tree =>



for level 2, the updated table is,

Temprature	Humidity	Sky	# Rain
Cool	High	Cloudy	Rain
Cool	Low	Cloudy	No Rain

The "Splitting Criteria" for level 2 will be



So, the updated tree is,