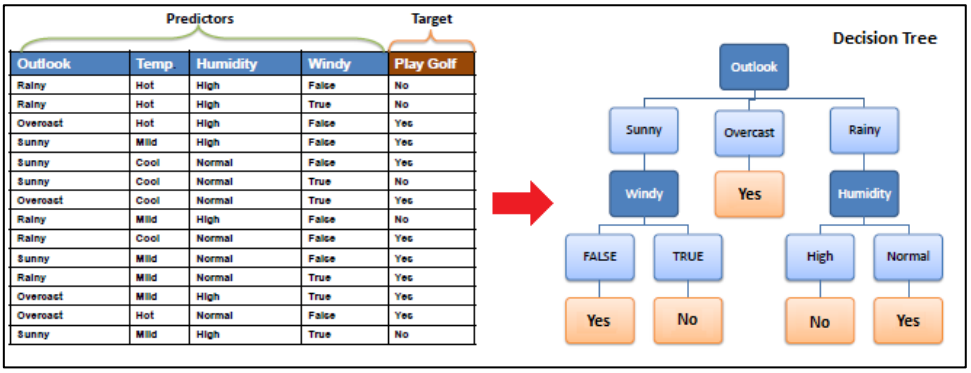
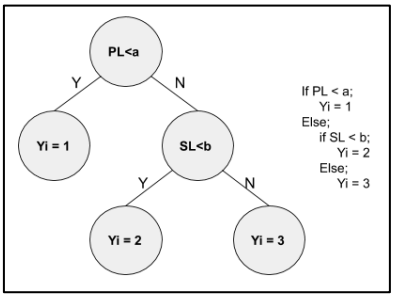
- **Geometric Intuition of decision tree: Axis parallel hyperplanes**



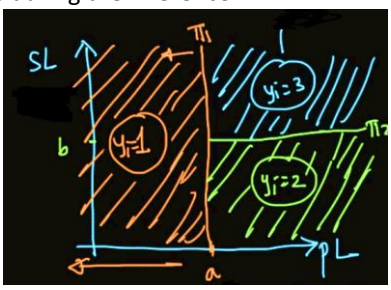
Predictors / Target

Outlook	Temp	Humidity	Windy	Play Golf
Rainy	Hot	High	False	No
Rainy	Hot	High	True	No
Overcast	Hot	High	False	Yes
Sunny	Mild	High	False	Yes
Sunny	Cool	Normal	False	Yes
Sunny	Cool	Normal	True	No
Overcast	Cool	Normal	True	Yes
Rainy	Mild	High	False	No
Rainy	Cool	Normal	False	Yes
Sunny	Mild	Normal	False	Yes
Rainy	Mild	Normal	True	Yes
Overcast	Mild	High	True	Yes
Overcast	Hot	Normal	False	Yes
Sunny	Mild	High	True	No

  - Decision tree is a flowchart like tree structure where
    - Each internal node denotes a test on an attribute
    - Each branch represents an outcome of the test
    - Each leaf node (terminal node) holds a class label.
  - We can alternatingly think that decision trees are a group of nested IF-ELSE conditions which can be modelled as a tree where the decision is made in the internal node and output is obtained in the leaf node.
  - Below is a simple decision tree of the IRIS Data Set. a & b are the sepal, petal lengths, respectively.



  - Geometrically, we can think of decision trees as a set of an axis-parallel hyperplanes that divide the space into the number of hyper cuboids during the inference.



  - we can see that in the Image Decision boundaries are axis parallel & have a decision surface for each leaf node (Yi).

- **Entropy**
  - Entropy is a measure of unpredictability. Feature with higher entropy has more random values than the one with lower entropy.
  - Suppose we tossed a coin 4 times,

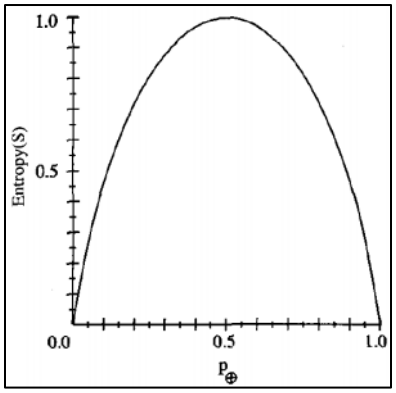
Output	P(H)	P(T)	Entropy	Interpretation
{H, H, T, T}	50%	50%	1	Output is a random event
{T, T, T, H,}	25%	75%	0.8113	The result is less random i.e., biased coin

  - For a given random variable Y which has K values, then enropy of Y is

$$H(y) = -\sum_{i=1}^{K} P(y_i).\log_b\left(P(y_i)\right)$$
$$P(y_i) = P(Y = y_i)$$
$$typ.b = 2(\lg) or\ b = e(ln)$$

  - E.g., for the above play golf example, there are two classes

$$H(y) = -P(y = yes) - P(y = no)$$
$$H(y) = -\frac{9}{14}\log\left(\frac{9}{14}\right) - \frac{5}{14}\log\left(\frac{5}{14}\right) = 0.94$$

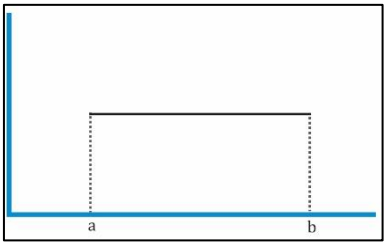


- Comparing entropies for different cases for 2-class example

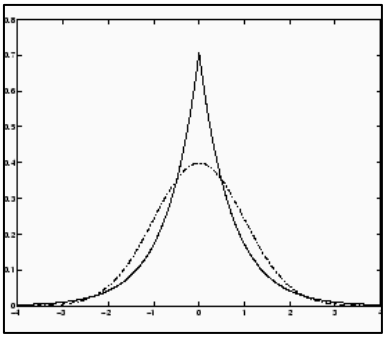
Sr. No.	P(Y+)	P(Y-)	Entropy
1	99%	1%	0.08
2	50%	50%	1.00
3	100%	0%	0.00

  - If both classes are equally probable, we have maximum entropy value of 1
  - If one class fully dominates, the entropy value becomes 0

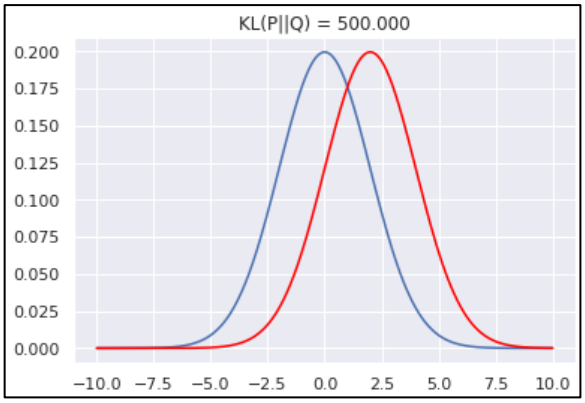
- **Comparing entropies for different cases for Multi-class example**
  - If all classes are equiprobable, the entropy will be maximum i.e., uniform distribution



  - If one class has 100% probability and others are zero, the entropy is minimum
  - In this figure entropy of skewed distribution will be lesser than that of normally distributed. More peaked is the distribution lesser will be the entropy



- **Kullback-Leibler Divergence or relative entropy**



  - KL Divergence measures the difference between two probability distributions over the same variable X.
  - For Discrete probability distributions P & Q on same probability space X

$$D_{KL}(P||Q) = \sum_{x \in X} p(x).\log\left(\frac{p(x)}{q(x)}\right)$$

  - For distributions P and Q of a continuous random variable

$$D_{KL}(P||Q) = \int_{-\infty}^{\infty} p(x).\log\left(\frac{p(x)}{q(x)}\right)dx$$

$$p(x)\ and\ q(x)\ are\ probability\ densities\ of\ P\ and\ Q$$

  - It has higher value for dissimilar or divergent distribution and less value for similar distributions

- **Information Gain**
  - The information gain is based on the decrease in entropy after a dataset is split on an attribute. Constructing a decision tree is all about finding the attribute that returns the highest information gain (i.e., the most homogeneous branches).
  - Example.
    - As we know entropy of golf example was 0.94

$$H(y) = -P(y = yes) - P(y = no)$$
$$H(y) = -\frac{9}{14}\log\left(\frac{9}{14}\right) - \frac{5}{14}\log\left(\frac{5}{14}\right) = 0.94$$

- The dataset is then split on the different attributes. The entropy for each branch is calculated. Then it is added proportionally, to get total entropy for the split. The resulting entropy is subtracted from the entropy before the split. The result is the Information Gain, or decrease in entropy.

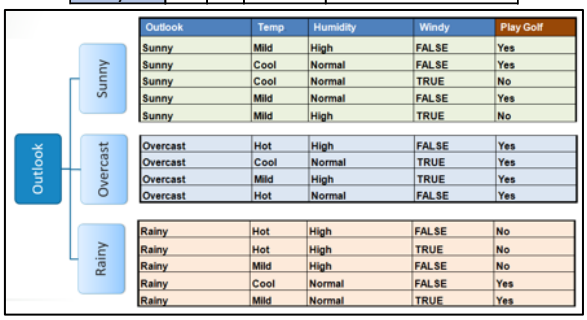
Outlook	Play Golf			W.E(Play, Outlook)	Temp	Play Golf			W.E(Play, Temp)
	Yes	No	Entropy			Yes	No	Entropy	
Sunny	3	2	0.9710	0.3468	Hot	2	2	1.0000	0.2857
Overcast	4	0	0.0000	0.0000	Mild	4	2	0.9183	0.3936
Rainy	2	3	0.9710	0.3468	Cool	3	1	0.8113	0.2318
Gain	0.246				Gain	0.029			
Humidity	Play Golf			W.E(Play, Windy)	Windy	Play Golf			W.E(Play, Temp)
	Yes	No	Entropy			Yes	No	Entropy	
High	3	4	0.9852	0.4926	FALSE	6	2	0.8113	0.4636
Normal	6	1	0.5917	0.2958	TRUE	3	3	1.0000	0.4286
Gain	0.152				Gain	0.048			

$$Gain(T, X) = Entropy(T) - W.Entropy(T, X)$$
$$G(Play\ Golf, Outlook) = Entropy(Play\ golf) - W.Entropy(Play\ Golf, Outlook)$$
$$= 0.94 - 0.693 = 0.247$$

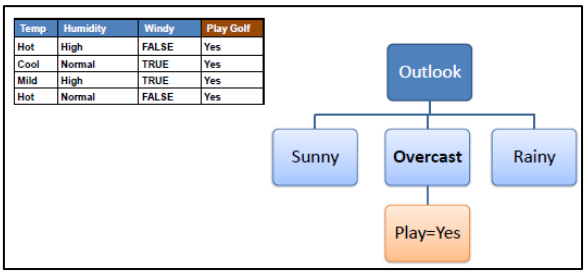
$$IG(y, D_i) = \sum_{i=1}^{n} \frac{|D_i|}{|D|} * H_{D_i}(y) - H_D(y)$$

- Choose attribute with the largest information gain as the decision node, divide the dataset by its branches and repeat the same process on every branch.

Outlook	Play Golf			W.E(Play,Outlook)
	Yes	No	Entropy	
Sunny	3	2	0.9710	0.3468
Overcast	4	0	0.0000	0.0000
Rainy	2	3	0.9710	0.3468



- A branch with entropy of 0 is a pure node there is no need to break it further

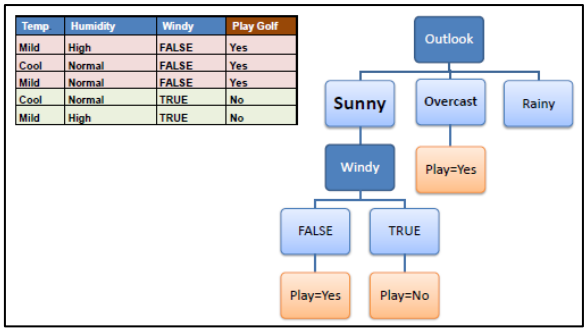


- A branch with entropy more than 0 needs further splitting.

Outlook	Temp	Humidity	Windy	Play Golf
Sunny	Mild	High	FALSE	Yes
Sunny	Cool	Normal	FALSE	Yes
Sunny	Cool	Normal	TRUE	No
Sunny	Mild	Normal	FALSE	Yes
Sunny	Mild	High	TRUE	No

Outlook	Feature	Temp	(Yes)	(No)
Sunny	Temp	Mild	2	1
		Cool	1	1
	Humidity	Normal	2	1
		High	1	1
	Windy	FALSE	3	0
		TRUE	0	2

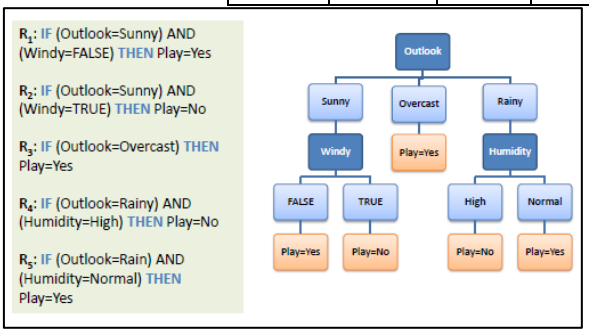
- Here, we can conclude that windy will be the important feature as it gives pure nodes



- Same thing can be applied to rainy and we can see that for rain, humidity gives pure nodes

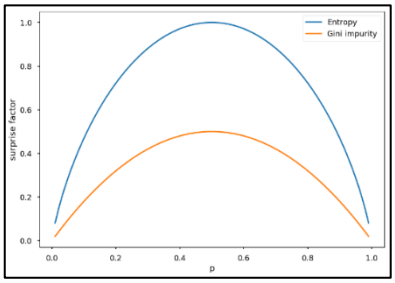
Outlook	Temp	Humidity	Windy	Play Golf
Rainy	Hot	High	FALSE	No
Rainy	Hot	High	TRUE	No
Rainy	Mild	High	FALSE	No
Rainy	Cool	Normal	FALSE	Yes
Rainy	Mild	Normal	TRUE	Yes

Outlook	Feature	Temp	(Yes)	(No)
Rainy	Temp	Hot	0	2
		Mild	1	1
		Cool	1	0
	Humidity	Normal	1	0
		High	0	3
	Windy	FALSE	1	2
		TRUE	1	1



  - We stop growing tree at following cases
    - When there is a pure node
    - There are very few points corresponding to a single class
    - If the tree is two deeper as depth of tree increases, we tend to overfit
    - In decision tree, our hyperparameter will be the depth of tree

- **Gini Impurity**
  - It is similar idea to entropy



$$Im_G(Y) = 1 - \sum_{i=1}^{k} (P(y_i)^2)$$
$$Y = y_1, y_2, y_3, \dots\dots, y_k$$

Sr. No.	P(Y+)	P(Y-)	Entropy	G.I.
1	99%	1%	0.08	0.02
2	50%	50%	1.00	0.50
3	100%	0%	0.00	0.00

  - Calculating logarithmic values is computational expensive that computing squares, thus as Entropy and Gini impurity behave similar to each other, Gini impurity is preferred for its compute efficiency.

- **Splitting numerical features**

Temp	85	80	83	70	68	65	64	72	69	75
Play	No	No	Yes	Yes	Yes	No	Yes	No	Yes	Yes

  - If we are having numerical features and then if we have to split the data.
  - We first sort them by numbers

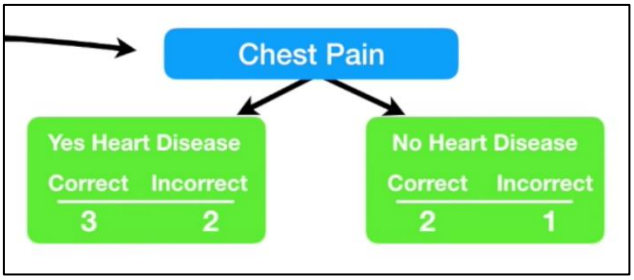
Temp	64	65	68	69	70	72	75	80	83	85
Play	Yes	No	Yes	Yes	Yes	No	Yes	No	Yes	No

  - We split the data at each threshold value such that D1 is dataset less than 65 & D2 will be the dataset more than 65. Same goes for every threshold value
  - And we calculate information gain for each of this dataset and then the combination giving maximum value of information gain will be chosen as classification node

- **Categorical features with many possible values**
  - Many times, we occur in such a situation where there are lot of categories e.g., pin codes. Here the list of pin codes will be bigger as each of them is representing an area but corresponding to it the y' will be less or more
  - We convert these features into numerical feature such that

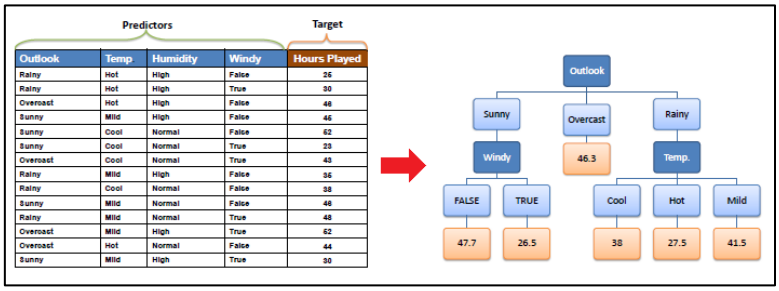
$$P\ (Y = 1\ |pin_j)\ = \frac{\#y = 1\ for\ pin_j}{\#pin_j}$$

  - Further we will divide the datasets on threshold values like we did in numerical features
- **Overfitting and underfitting**
  - As depth of the tree increases, possibility of having very few points at lead node also increases.
  - Also, these very few points are most likely to be noisy points. If we are considering them in our decision tree. We are basically overfitting
  - As the depth of model increases interpretability of model also decreases. i.e., it's very hard to understand the model if there are multiple nested conditions.
  - Decision stump is a decision tree with only one branch. We predict the class which has majority of points



  - If depth of the model is less, the model will be underfit.
  - Here depth will be a hyperparameter which needs to be tuned based on cross-validation.

- **Train and Run time complexity**
  - Train time complexity = O(n.log(n)d)
  - Run time space complexity = O(nodes)
  - Typically, depth of trains = 5 or 10. Else it become shard to interpret
  - Run time complexity = O(depth)
  - Decision Trees are suitable: Large data, small dimensionality, low latency requirement
- **Regression using Decision Trees**



  - **Approach1: - By standard deviation**
    - Standard deviation for **one** attribute:
    - Standard Deviation (**S**) is for tree building (branching).
    - Coefficient of Deviation (**CV**) is used to decide when to stop branching. We can use Count (**n**) as well.
    - Average (**Avg**) is the value in the leaf nodes.

Hours Played
25
30
46
45
52
23
43
35
38
46
48
52
44
30

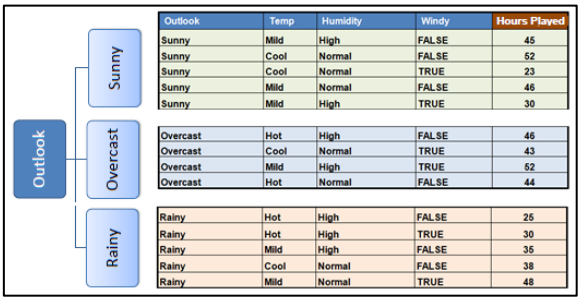
$$\boldsymbol{Count} = n = 14$$

$$\boldsymbol{Average} = \bar{x} = \frac{\sum x}{n} = 39.8$$

$$\boldsymbol{Standard\ Deviation} = S = \sqrt{\frac{\sum(x-\bar{x})^2}{n}} = 9.32$$

$$\boldsymbol{Coeffeicient\ of\ Variation} = CV = \frac{S}{\bar{x}} * 100\% = 23\%$$

- Standard deviation for **two** attributes (target and predictor):
- The standard deviation reduction is based on the decrease in standard deviation after a dataset is split on an attribute. Constructing a decision tree is all about finding attribute that returns the highest standard deviation reduction (i.e., the most homogeneous branches).



Outlook	Temp	Humidity	Windy	Hours Played
Sunny	Mild	High	FALSE	45
Sunny	Cool	Normal	FALSE	52
Sunny	Cool	Normal	TRUE	23
Sunny	Mild	Normal	FALSE	46
Sunny	Mild	High	TRUE	30
Overcast	Hot	High	FALSE	46
Overcast	Cool	Normal	TRUE	43
Overcast	Mild	High	TRUE	52
Overcast	Hot	Normal	FALSE	44
Rainy	Hot	High	FALSE	25
Rainy	Hot	High	TRUE	30
Rainy	Mild	High	FALSE	35
Rainy	Cool	Normal	FALSE	38
Rainy	Mild	Normal	TRUE	48

		Hours Played (StDev)	Count
Outlook	Overcast	3.49	4
	Rainy	7.78	5
	Sunny	10.87	5
			14

- Step 1: The standard deviation of the target is calculated.

  Standard deviation (Hours Played) = 9.32
- Step 2: The dataset is then split on the different attributes. The standard deviation for each branch is calculated. The resulting standard deviation is subtracted from the standard deviation before the split. The result is the SDR (standard deviation reduction)

		Hours Played (StDev)
Outlook	Overcast	3.49
	Rainy	7.78
	Sunny	10.87
SDR=1.66		

		Hours Played (StDev)
Temp.	Cool	10.51
	Hot	8.95
	Mild	7.65
SDR=0.17		

		Hours Played (StDev)
Humidity	High	9.36
	Normal	8.37
SDR=0.28		

		Hours Played (StDev)
Windy	False	7.87
	True	10.59
SDR=0.29		

- Step 3: The attribute with the largest standard deviation reduction is chosen for the decision node.

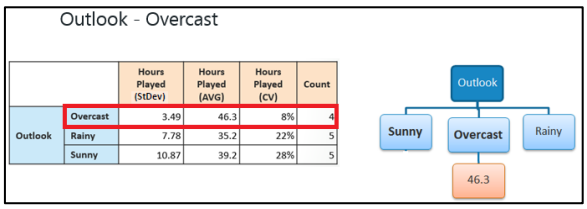
$$SDR(T,X) = S(T) - S(T,X)$$

SDR(Hours , Outlook) = **S**(Hours) – **S**(Hours, Outlook)

= 9.32 – 7.66 = 1.66

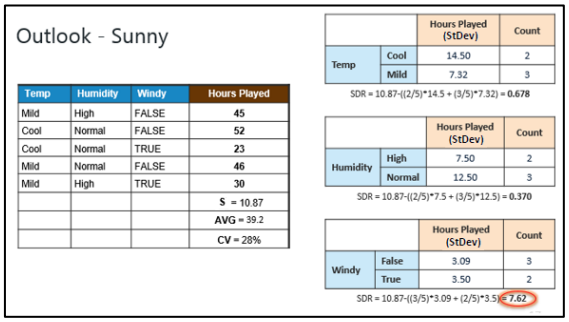
★		Hours Played (StDev)
Outlook	Overcast	3.49
	Rainy	7.78
	Sunny	10.87
SDR=1.66		

- Step 4a: The dataset is divided based on the values of the selected attribute. This process is run recursively on the non-leaf branches, until all data is processed.
- In practice, we need some termination criteria. For example, when coefficient of variation (CV) for a branch becomes smaller than a certain threshold (e.g., 10%) and/or when too few instances (n) remain in the branch (e.g., 3).
- Step 4b: "Overcast" subset does not need any further splitting because its CV (8%) is less than the threshold (10%). The related leaf node gets the average of the "Overcast" subset.



		Hours Played (StDev)	Hours Played (AVG)	Hours Played (CV)	Count
Outlook	Overcast	3.49	46.3	8%	4
	Rainy	7.78	35.2	22%	5
	Sunny	10.87	39.2	28%	5

- Step 4c: However, the "Sunny" branch has an CV (28%) more than the threshold (10%) which needs further splitting. We select "Windy" as the best node after "Outlook" because it has the largest SDR.



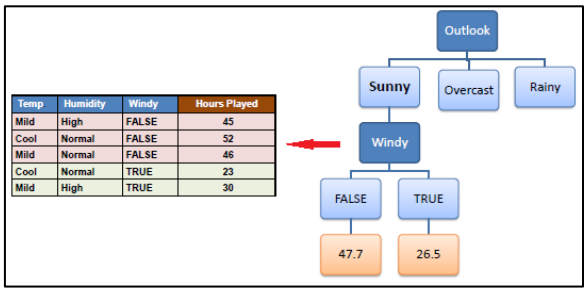
Temp	Humidity	Windy	Hours Played
Mild	High	FALSE	45
Cool	Normal	FALSE	52
Cool	Normal	TRUE	23
Mild	Normal	FALSE	46
Mild	High	TRUE	30
			S = 10.87
			AVG = 39.2
			CV = 28%

		Hours Played (StDev)	Count
Temp	Cool	14.50	2
	Mild	7.32	3

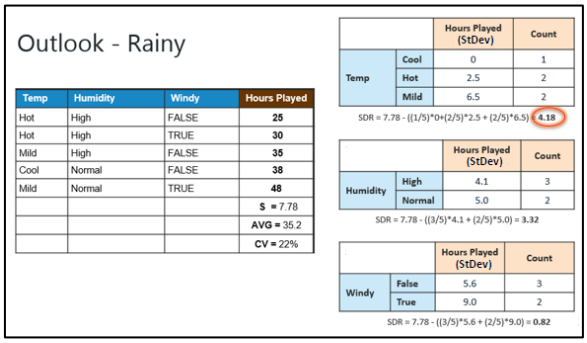
		Hours Played (StDev)	Count
Humidity	High	7.50	2
	Normal	12.50	3

		Hours Played (StDev)	Count
Windy	False	3.09	3
	True	3.50	2

Because the number of data points for both branches (FALSE and TRUE) is equal or less than 3 we stop further branching and assign the average of each branch to the related leaf node.



- Step 4d: Moreover, the "rainy" branch has an CV (22%) which is more than the threshold (10%). This branch needs further splitting. We select "Temp" as the best node because it has the largest SDR.



- Because the number of data points for all three branches (Cool, Hot and Mild) is equal or less than 3 we stop further branching and assign the average of each branch to the related leaf node.