Machine Learning with Python

Decision Trees

- A supervised learning predictive model that uses a set of binary rules to calculate a target value

- Used for either classification(categorical variables) or regression(continuous variables)

- In this method, the given data is split into one or more homogeneous sets based on most significant input variable

- Tree generating algorithm determines

  - Which variable to split at a node

  - Decision to stop or make a split again

  - Assign terminal nodes to a class

Decision Trees

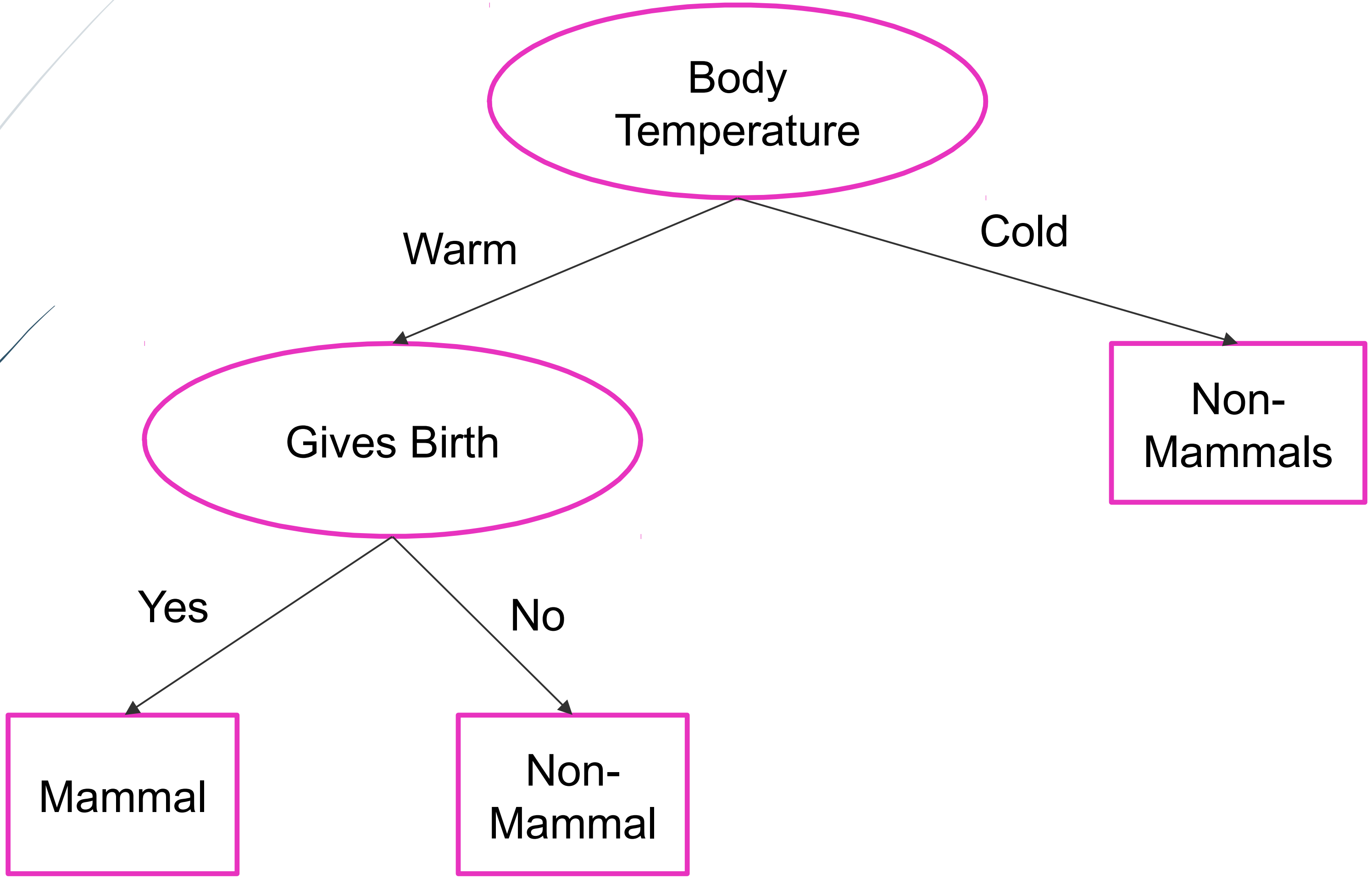
Sno	Species Name	Body_Temp	Gives_Birth	Type
1	SP101	Cold-Blooded	No	Non-Mammal
2	SP102	Warm-Blodded	No	Non-Mammal
3	SP103	Cold-Blooded	No	Non-Mammal
4	SP104	Warm-Blodded	Yes	Mammal
5	SP105	Warm-Blodded	No	Non-Mammal
6	SP106	Warm-Blodded	No	Non-Mammal
7	SP107	Warm-Blodded	Yes	Mammal
8	SP108	Cold-Blooded	No	Non-Mammal
9	SP109	Cold-Blooded	No	Non-Mammal
10	SP110	Warm-Blodded	Yes	Mammal
11	SP111	Warm-Blodded	Yes	Mammal
12	SP112	Warm-Blodded	No	Non-Mammal
13	SP113	Cold-Blooded	No	Non-Mammal
14	SP114	Warm-Blodded	Yes	Mammal

Decision Trees

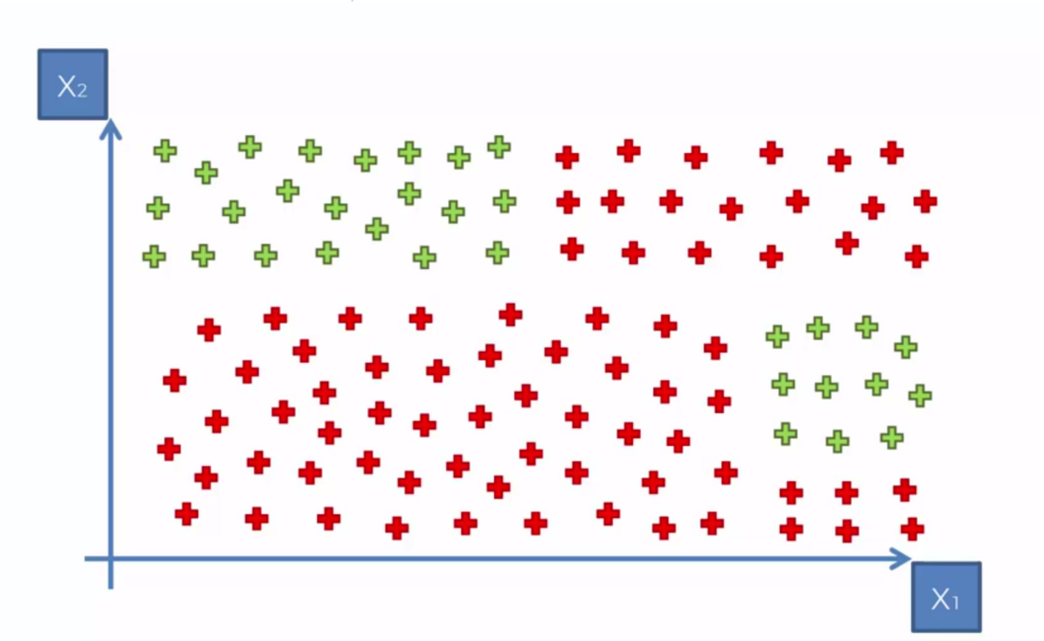
The decision tree will be created as follows:

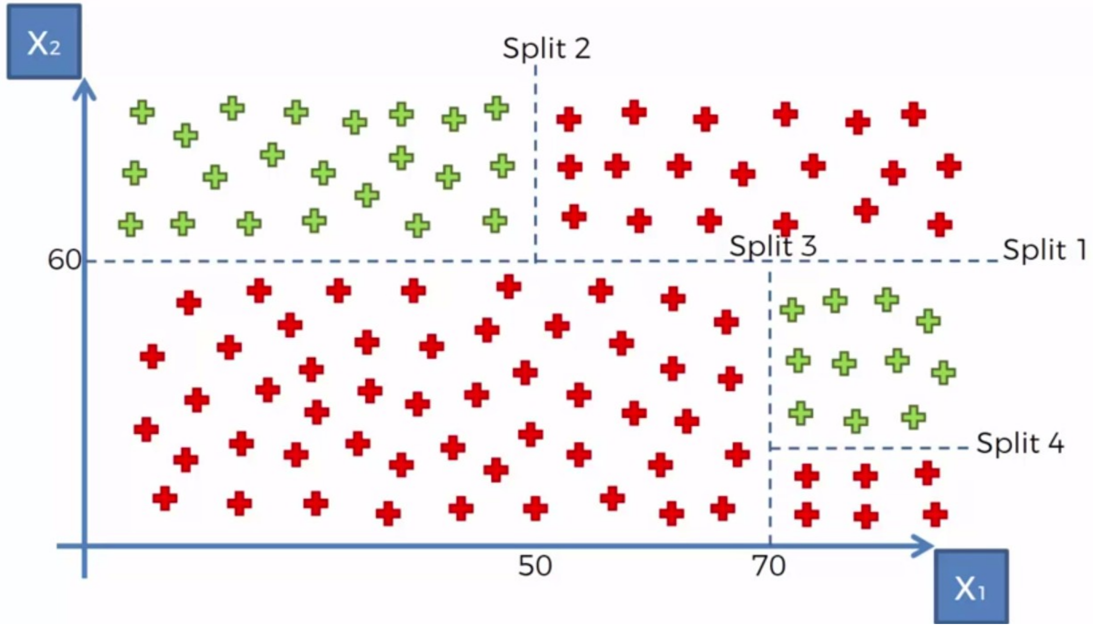
1. The splits will be done to partition the data into purer subsets. So, when the split is done on Body Temperature, all the species that are cold-blooded are found to be “non-mammals”. So when the Body Temperature is “Cold- Blooded”, we get a purest subset where all the species belong to “non-Mammals”.
2. But for “warm-blooded” species, we still have both types present: mammals and non-mammals. So, now the algorithm splits on the values of Gives_Birth. This time, when the values are “Yes”, we get all type values as “Mammals” and when the values are “No”, we get all type values as “Non-Mammals”.
3. The nodes represents Attributes while the edges represent values.

Decision Tree – An Example

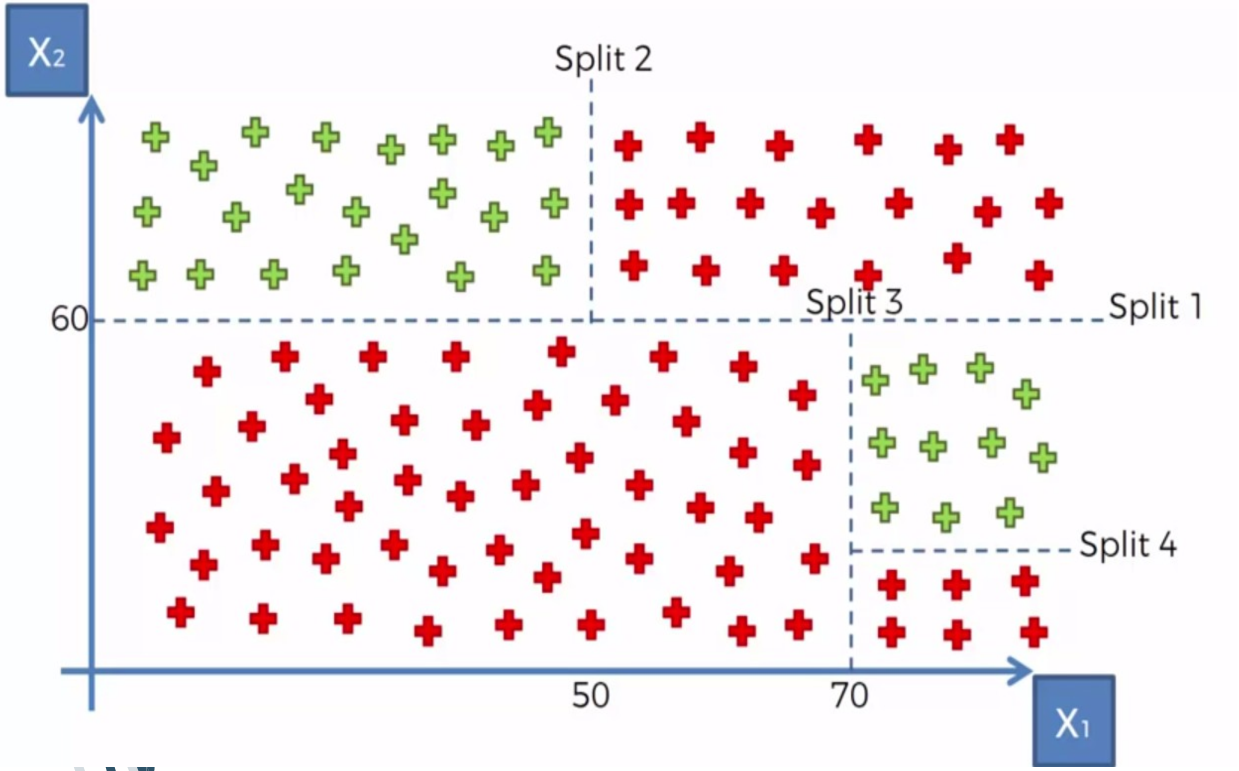


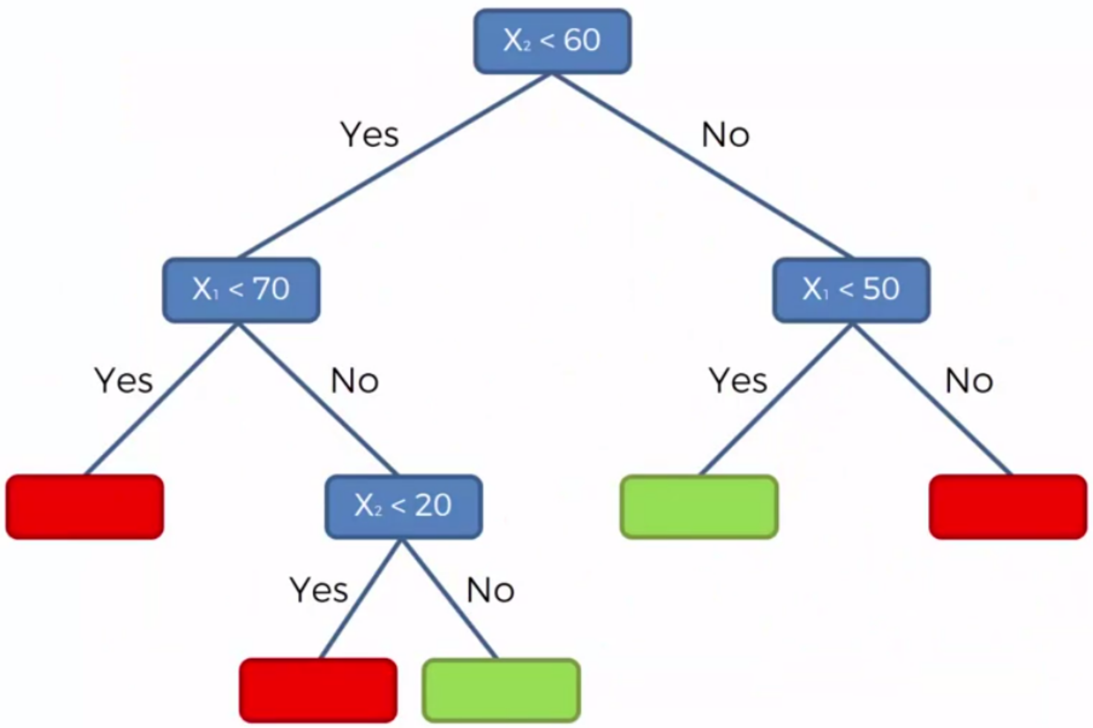
Decision Tree – Another Example





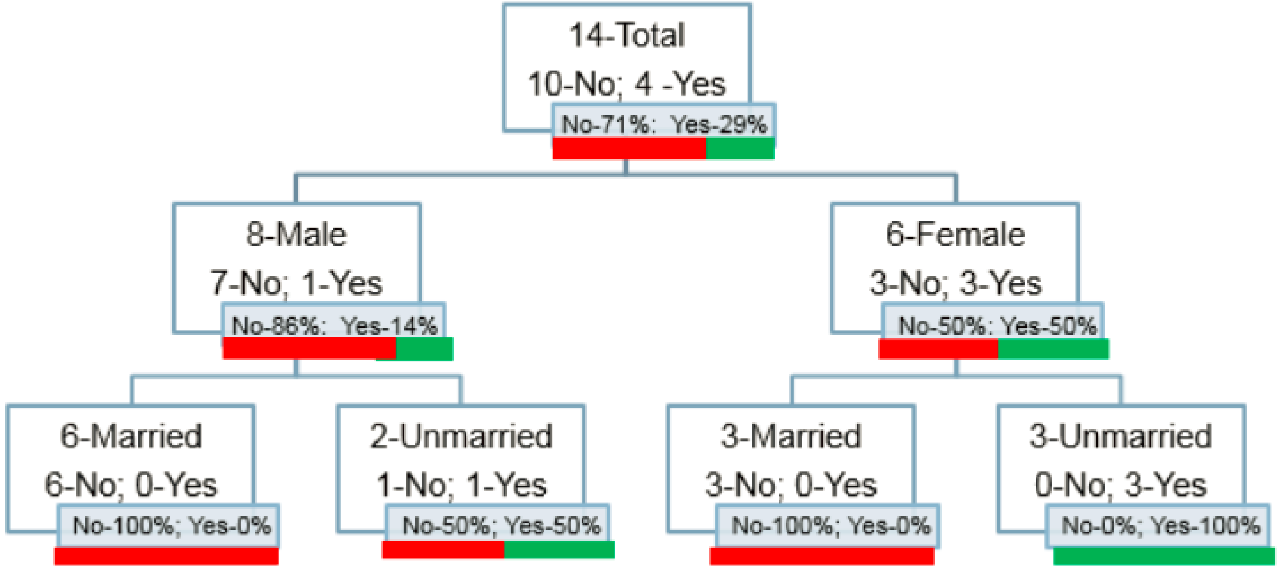
Decision Tree – Another Example





Decision Tree – Another Example

Sr No	Gender	Marital Status	Ordered the product
1	M	Married	No
2	F	Unmarried	Yes
3	M	Married	No
4	M	Married	No
5	M	Married	No
6	M	Married	No
7	F	Unmarried	Yes
8	M	Unmarried	Yes
9	F	Married	No
10	M	Married	No
11	F	Married	No
12	M	Unmarried	No
13	F	Married	No
14	F	Unmarried	Yes



14-Total
10-No; 4 -Yes
No-71%: Yes-29%

8-Male
7-No; 1-Yes
No-86%: Yes-14%

6-Female
3-No; 3-Yes
No-50%: Yes-50%

6-Married
6-No; 0-Yes
No-100%; Yes-0%

2-Unmarried
1-No; 1-Yes
No-50%; Yes-50%

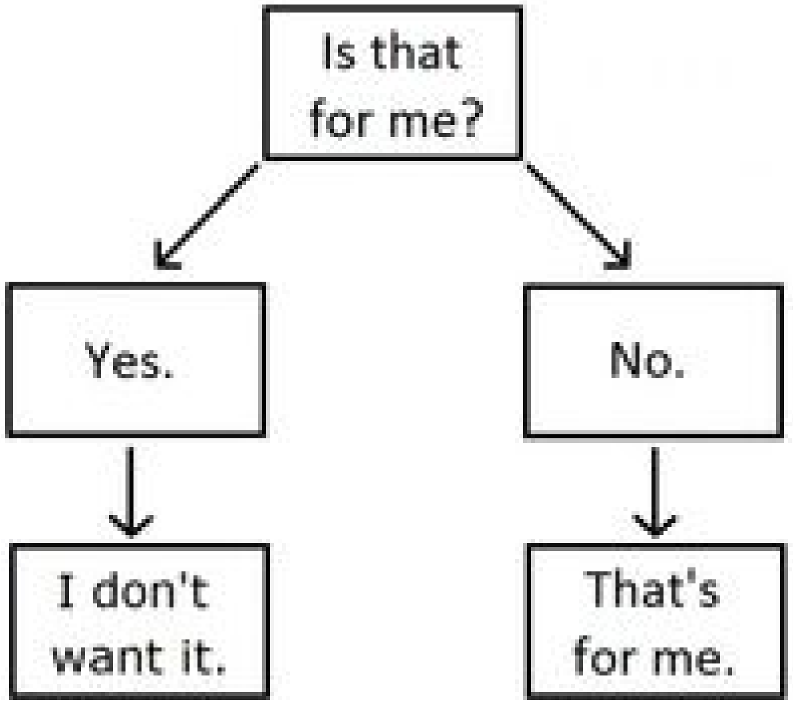
3-Married
3-No; 0-Yes
No-100%; Yes-0%

3-Unmarried
0-No; 3-Yes
No-0%; Yes-100%

Gender	Marital Status	Product order
M	Married	??
F	Unmarried	??

How do Children and cats use DT?

Children's Decision-Making Tree



Is that for me?

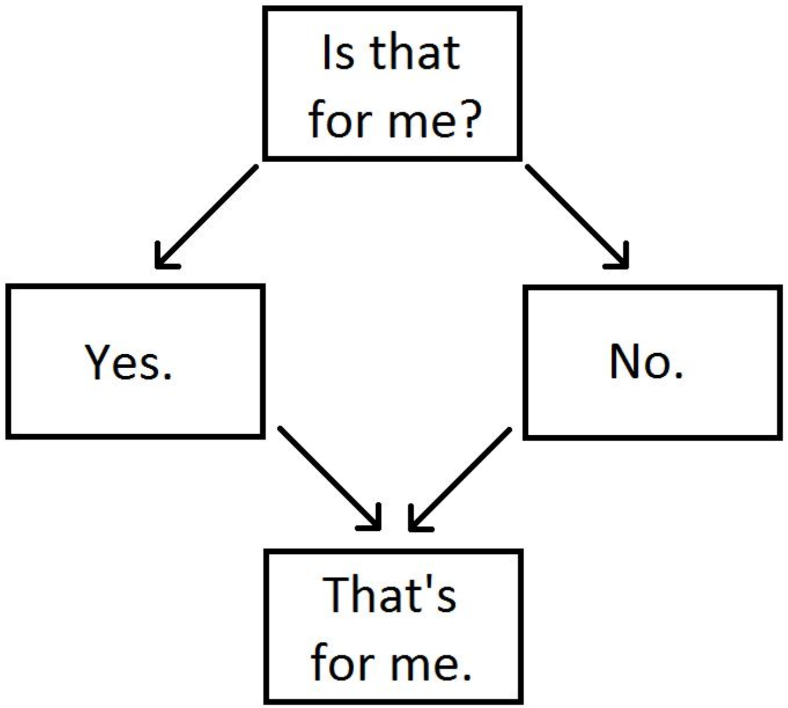
Yes.

No.

I don't want it.

That's for me.

My Cat's Decision-Making Tree.



Is that for me?

Yes.

No.

That's for me.

Measures Used for Split

- Gini Index
- Entropy
- Information Gain

Gini

Gini Index: It is the measure of inequality of distribution. It says if we select two items from a population at random then they must be of same class and probability for this is 1 if population is pure.

- It works with categorical target variable “Success” or “Failure”.
- It performs only Binary splits
- Lower the value of Gini, higher the homogeneity.
- CART (Classification and Regression Tree) uses Gini method to create binary splits.

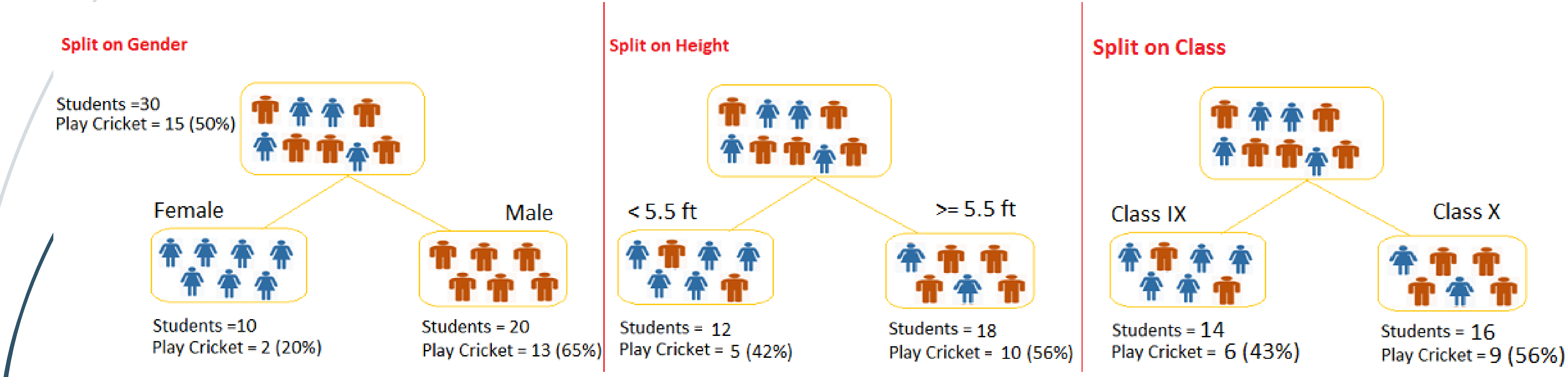
Process to calculate Gini Measure:

$$Gini = 1 - \sum_{j} p_j^2$$

P(j) is the Probability of Class j

Split Example

Let’s say we have a sample of 30 students with three variables Gender (Boy/ Girl), Class(IX/ X) and Height (5 to 6 ft). 15 out of these 30 play cricket in leisure time. Now, I want to create a model to predict who will play cricket during leisure period?



Entropy

Entropy is a way to measure impurity.

Less impure node requires less information to describe it and more impure node requires more information. If the sample is completely homogeneous, then the entropy is zero and if the sample is an equally divided it has entropy of one.

$$E(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

$$E(T,X) = \sum_{c \in X} P(c)E(c)$$

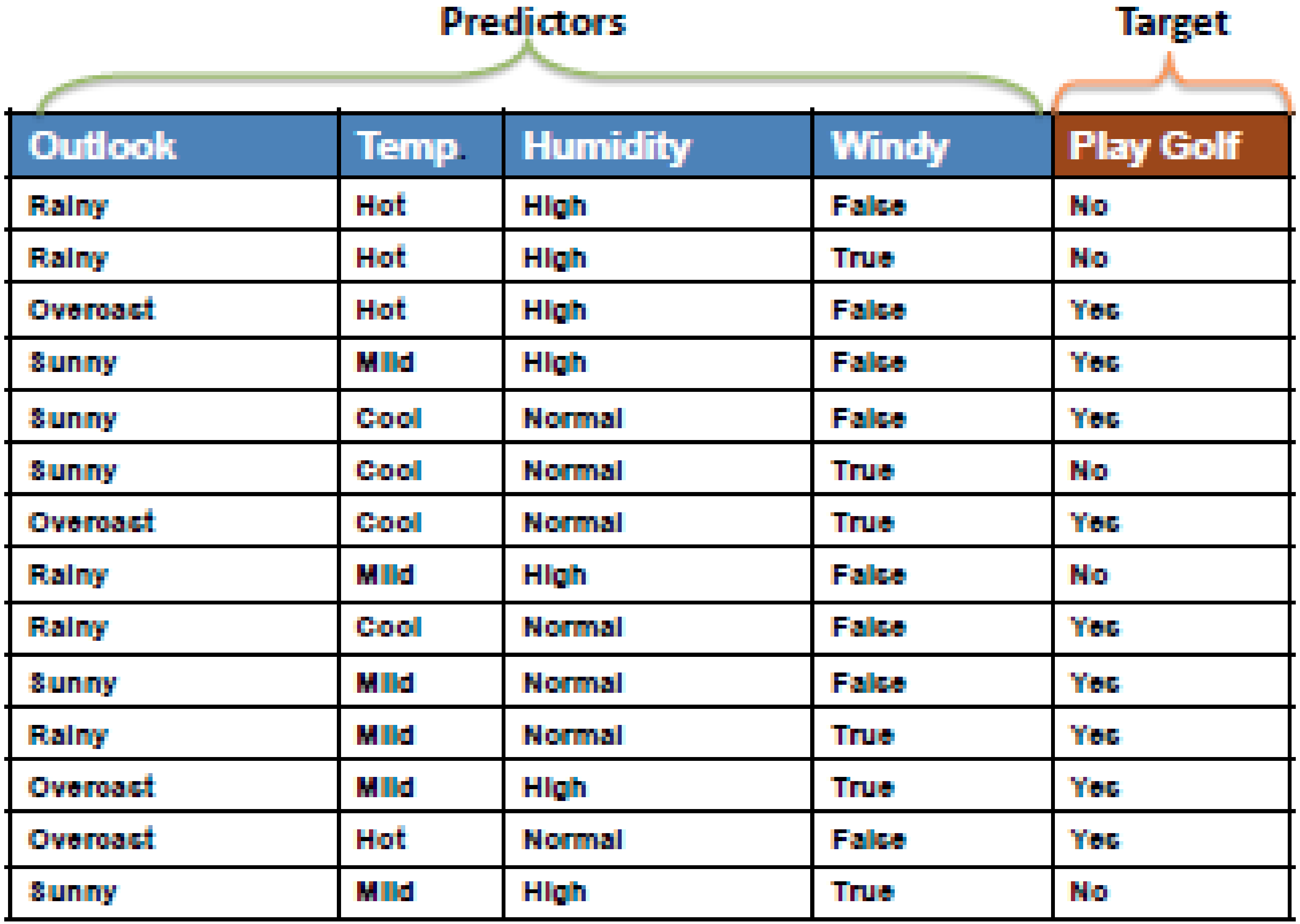
Information Gain

Information Gain is simply a mathematical way to capture the amount of information one gains(or reduction in randomness) by picking a particular attribute

In a decision algorithm, we start at the tree root and split the data on the feature that results in the largest **information gain (IG)**. In other words, IG tells us how important a given attribute is.

The **Information Gain (IG)** can be defined as follows:

$$Gain(T,X) = Entropy(T) - Entropy(T,X)$$



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

Play Golf	
Yes	No
9	5

Entropy(PlayGolf) = Entropy (5,9)

= Entropy (0.36, 0.64)

= - (0.36 $\log_2$ 0.36) - (0.64 $\log_2$ 0.64)

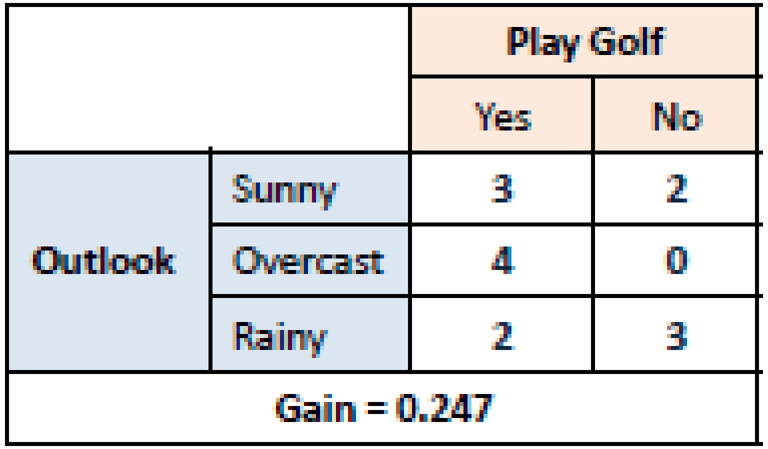
= 0.94

		Play Golf		
		Yes	No	
Outlook	Sunny	3	2	5
	Overcast	4	0	4
	Rainy	2	3	5
				14

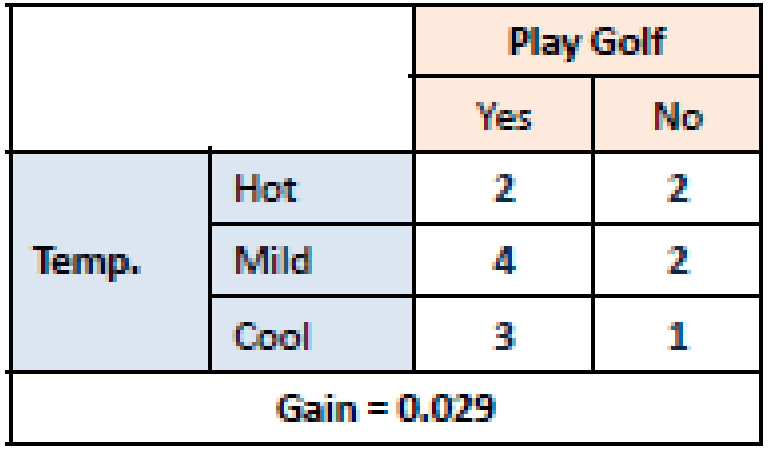
$$\mathbf{E}(\text{PlayGolf, Outlook}) = \mathbf{P}(\text{Sunny})*\mathbf{E}(3,2) + \mathbf{P}(\text{Overcast})*\mathbf{E}(4,0) + \mathbf{P}(\text{Rainy})*\mathbf{E}(2,3)$$

$$= (5/14)*0.971 + (4/14)*0.0 + (5/14)*0.971$$

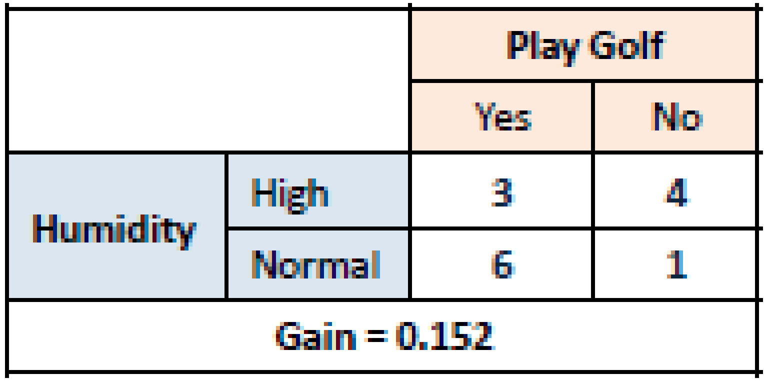
$$= 0.693$$



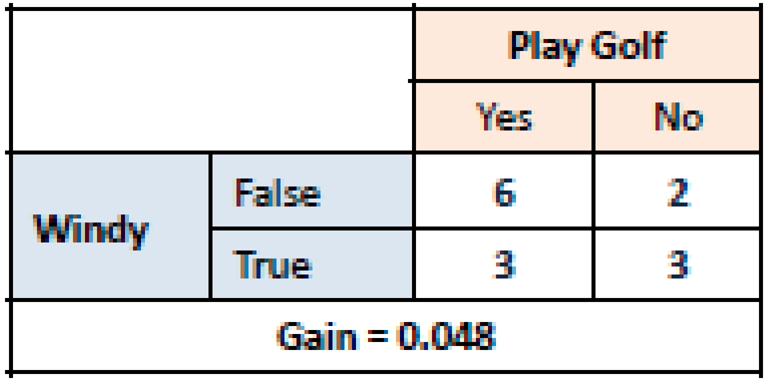
		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			



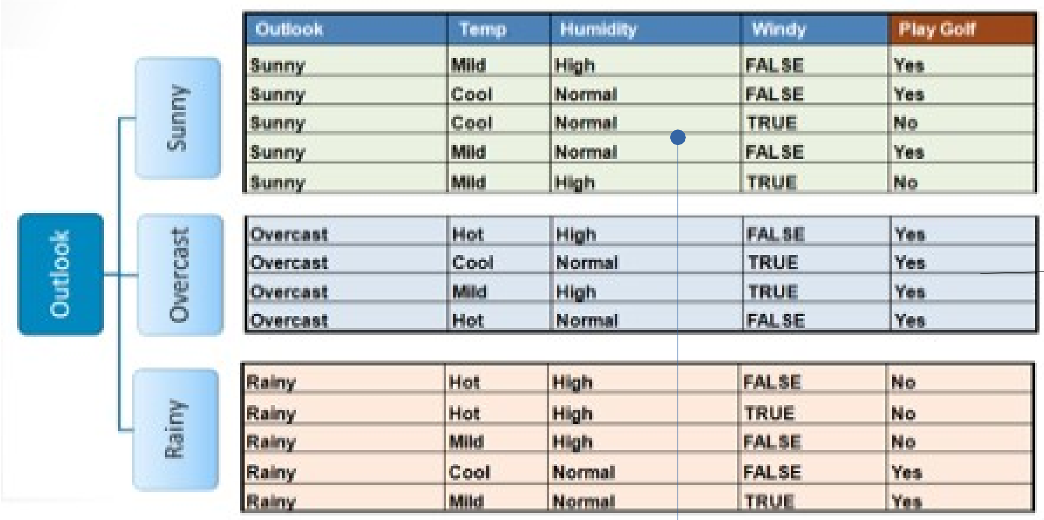
		Play Golf	
		Yes	No
Temp.	Hot	2	2
	Mild	4	2
	Cool	3	1
Gain = 0.029			



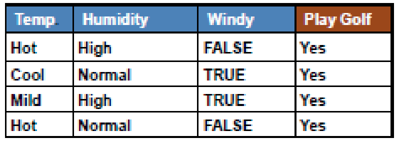
		Play Golf	
		Yes	No
Humidity	High	3	4
	Normal	6	1
Gain = 0.152			



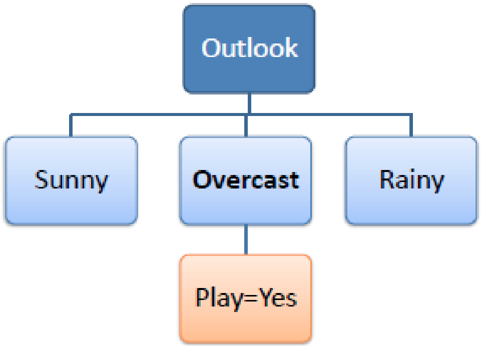
		Play Golf	
		Yes	No
Windy	False	6	2
	True	3	3
Gain = 0.048			

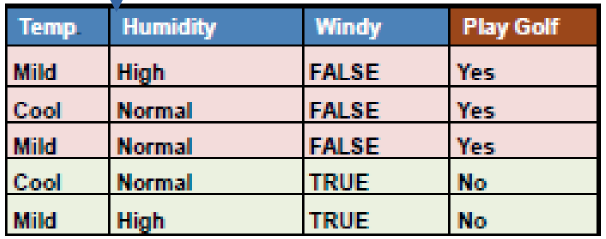


Outlook	Temp	Humidity	Windy	Play Golf
Sunny	Mild	High	FALSE	Yes
Sunny	Cool	Normal	FALSE	Yes
Sunny	Cool	Normal	TRUE	No
Sunny	Mild	Normal	FALSE	Yes
Sunny	Mild	High	TRUE	No
Overcast	Hot	High	FALSE	Yes
Overcast	Cool	Normal	TRUE	Yes
Overcast	Mild	High	TRUE	Yes
Overcast	Hot	Normal	FALSE	Yes
Rainy	Hot	High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

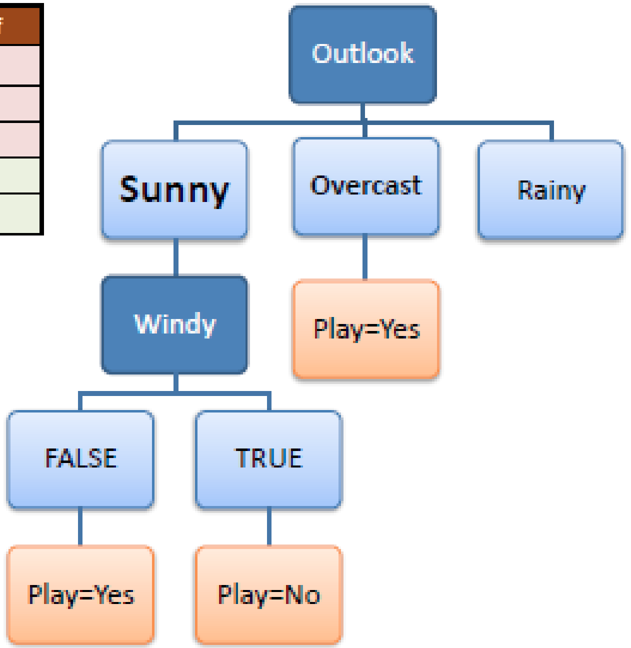


Temp	Humidity	Windy	Play Golf
Hot	High	FALSE	Yes
Cool	Normal	TRUE	Yes
Mild	High	TRUE	Yes
Hot	Normal	FALSE	Yes





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



Pros and Cons of Decision Trees

Advantages of Decision Trees:

- Its very interpretable and hence easy to understand
- It can be used to identify the most significant variables in your data-set

Disadvantages:

The model has very high chances of “over-fitting”

Avoiding Overfitting in Decision Trees

- Overfitting is the key challenge in case of Decision Trees.
- If no limit is set, in the worst case, it will end up putting each observation into a leaf node.
- It can be avoided by setting Constraints on Tree Size

Wisdom of crowd

What's everyone looking at?
I dunno. If it's something good, though, I don't wanna miss it.

Ensemble

Ensembling: Ensembling is a process of combining the results of multiple models to solve a given prediction or classification problem.

The three most popular methods for combining the predictions from different models are:

Bagging: Building multiple models (typically of the same type) from different subsamples of the training dataset. A limitation of bagging is that the same greedy algorithm is used to create each tree, meaning that it is likely that the same or very similar split points will be chosen in each tree making the different trees very similar (trees will be correlated). This, in turn, makes their predictions similar. We can force the decision trees to be different by limiting the features that the greedy algorithm can evaluate at each split point when creating the tree. This is called the **Random Forest** algorithm.

Boosting: Building multiple models (typically of the same type) each of which learns to fix the prediction errors of a prior model in the sequence of models.

Voting: Building multiple models (typically of differing types) and simple statistics (like calculating the mean) are used to combine predictions.

Voting Classifier – Majority/Hard Voting

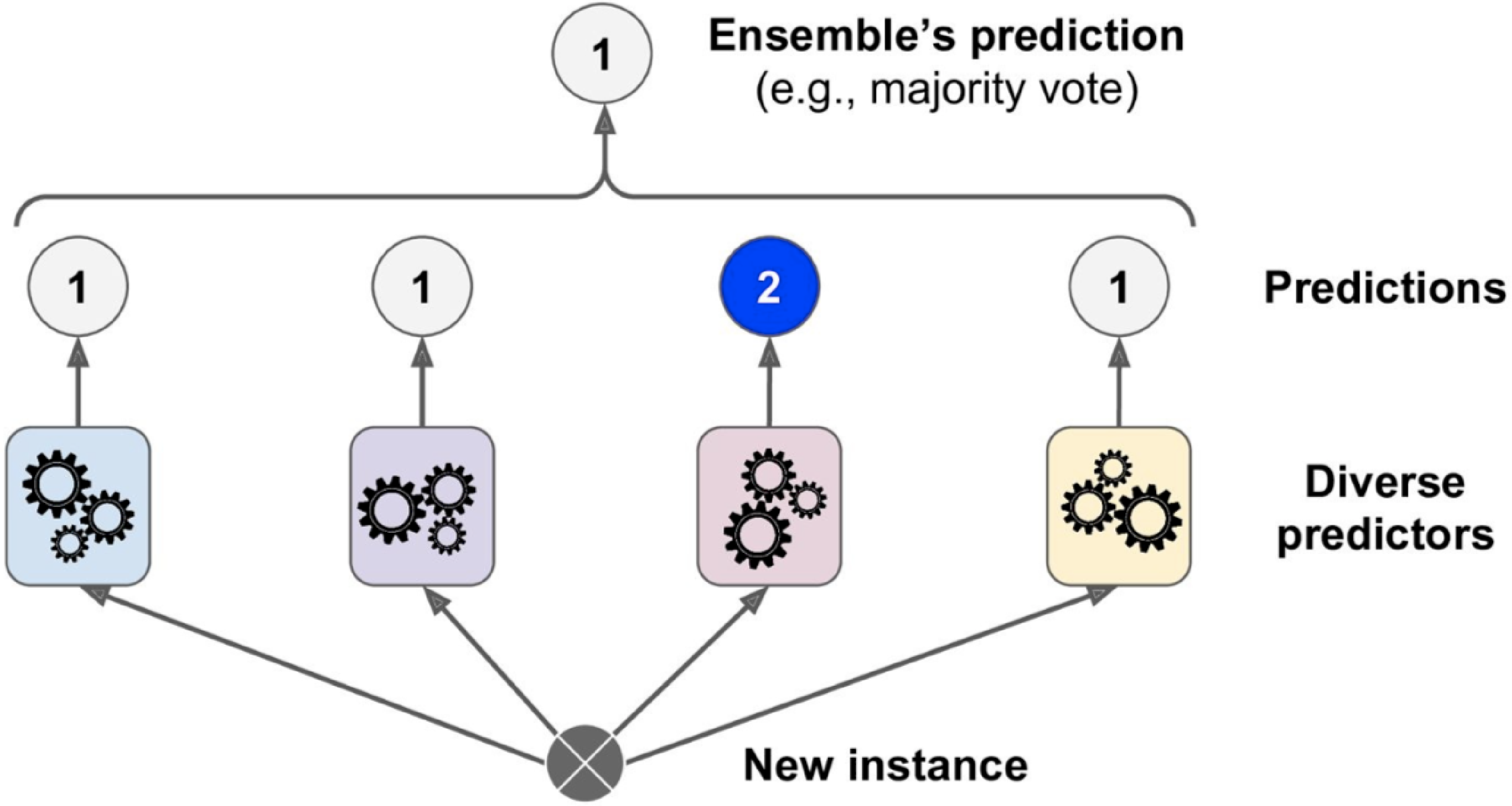
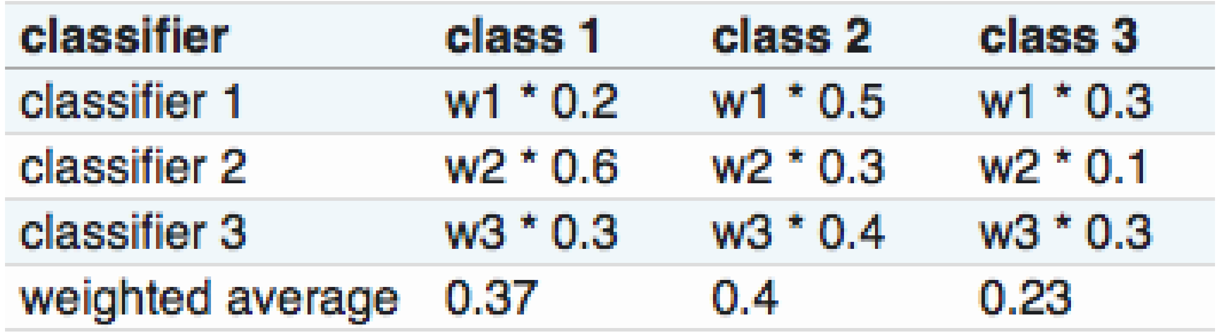


Figure 7-2. Hard voting classifier predictions

Voting Classifier – Soft Voting



classifier	class 1	class 2	class 3
classifier 1	w1 * 0.2	w1 * 0.5	w1 * 0.3
classifier 2	w2 * 0.6	w2 * 0.3	w2 * 0.1
classifier 3	w3 * 0.3	w3 * 0.4	w3 * 0.3
weighted average	0.37	0.4	0.23

- Specific weights can be assigned to each classifier via the weights parameter.
- When weights are provided, the predicted class probabilities for each classifier are collected, multiplied by the classifier weight, and averaged.
- The final class label is then derived from the class label with the highest average probability.

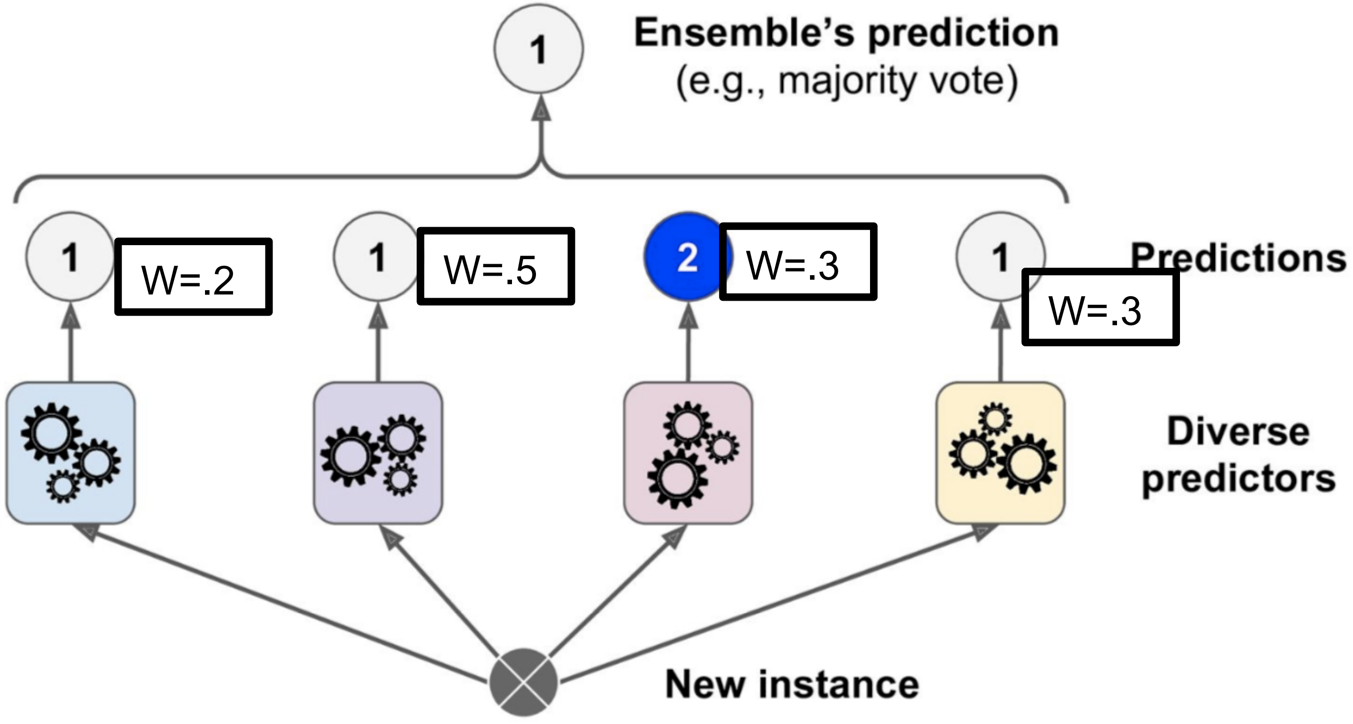
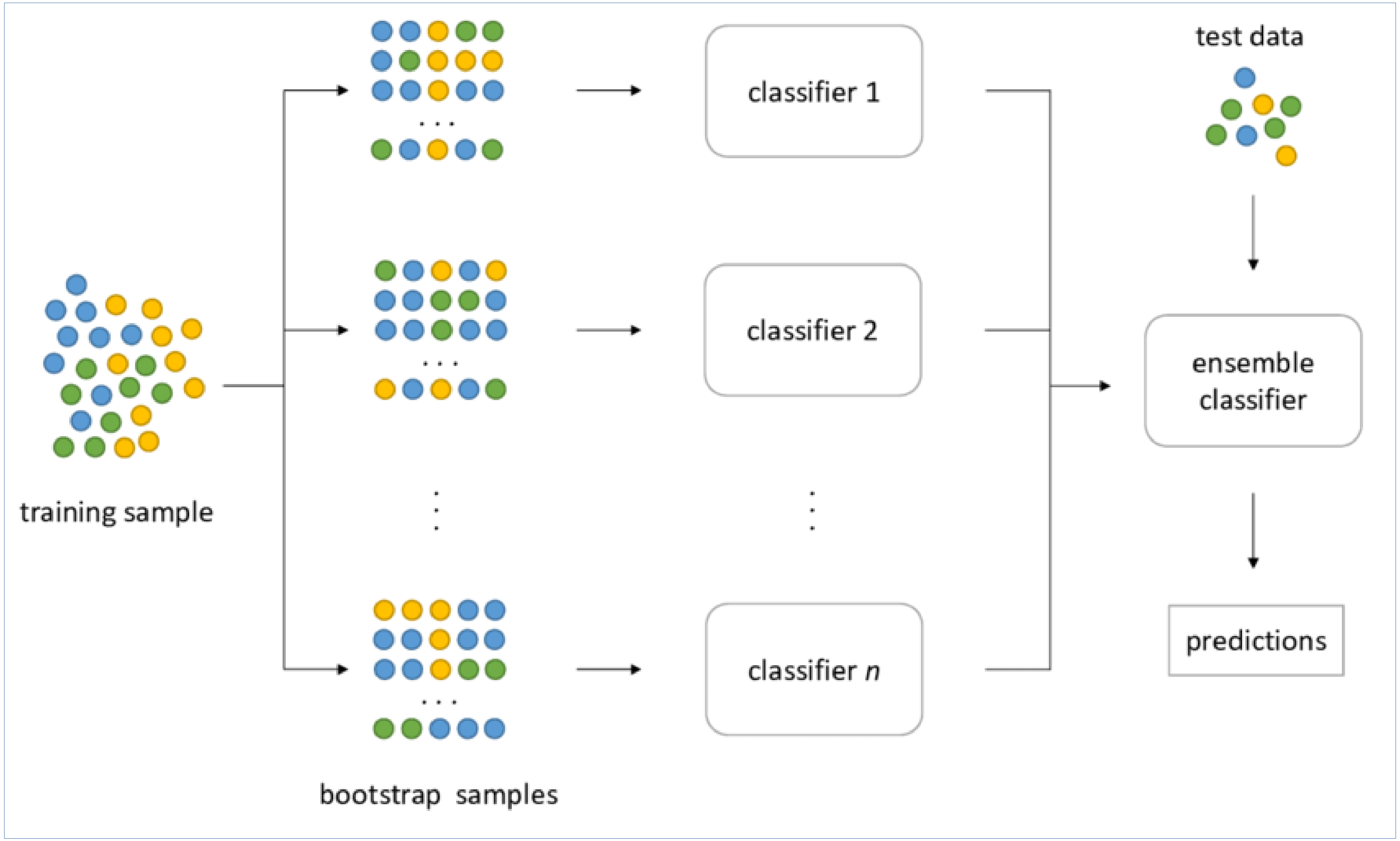


Figure 7-2. Hard voting classifier predictions

Bagging & Pasting

- Bagging is when sampling is done with replacement (WR).
- Pasting is when sampling is done without replacement(WOR).
- Ensemble make prediction by simply aggregating the prediction of all predictors(just like hard voting).
- Generally the net result is that ensemble has similar bias but lower variance



Bagging(WR) – bootstrap = True | Pasting(WOR) – bootstrap = False

Bagging performs best with algorithms that have high variance. A popular example are decision trees, often constructed without pruning.

Random Forests

Random Forests: Random Forests is an ensemble(bagging) modeling technique that works by building large number of decision tress. It takes a subset of observations and a subset of variables to build several decision trees. Each tree will play a role in determining the final outcome.

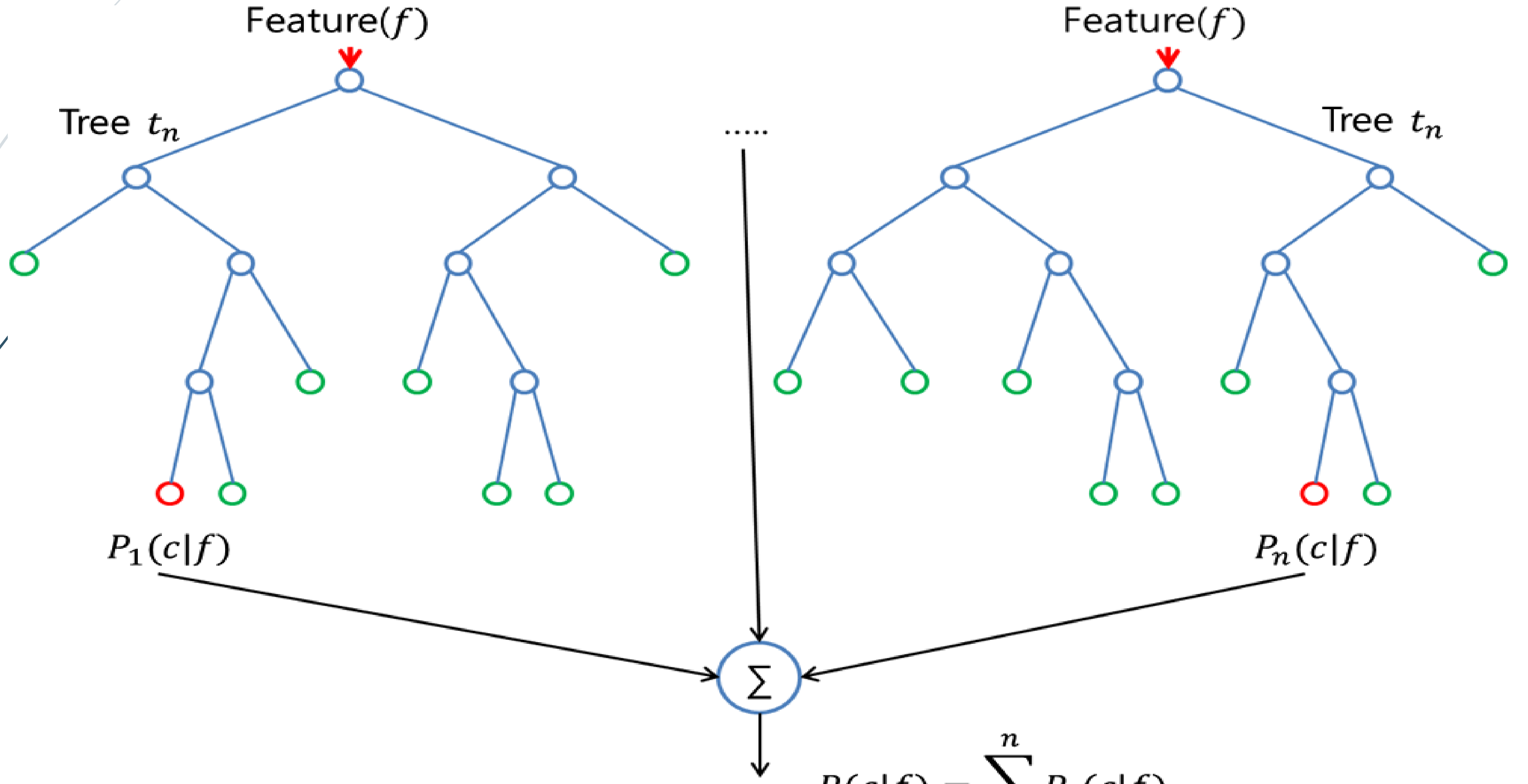
A random forest builds many trees. In forming each tree, it randomly selects the independent variables. This means that data used as training data for each tree is selected randomly with replacement. Each tree will give its output. The final output will be a function of each individual tree output.

The function can be:

- Mean of individual probability outcomes
- Vote Frequency

The important parameters used while using RandomForestClassifier in Python are:
n_estimators is used to fix number of trees in Random Forest.
max_depth - parameter determining when the splitting up of the decision tree stops.
min_samples_split - parameter monitoring the amount of observations in a bucket

Random Forests



$$P(c|f) = \sum_{1}^{n} P_n(c|f)$$

An Example

Lets say we have to decide whether a cricket player is Good or Bad. We can have the as "average runs scored" against various variables like "Opposition Played", "Country Played", "Matches Won", "Matches Lost" and many others. We can use a decision tree which splits on only certain attributes, to determine whether this player is Good or Bad. But we can improve our prediction, if we build a number of decision trees which split on different combination of variables. Then using the outcome of each tree, we can decide upon our final classification. This will greatly improve the performance which we get using a single tree. Another example can be the judgements delivered by Supreme Court. Supreme Court usually employs a bench of judges to deliver a judgement. Each judge gives its own judgment and then the outcome which gets the maximum voting becomes the final judgement.

An Example

Suppose, you want to watch a movie ‘X’ . You ask your friend ‘Amar’ if you should watch this movie.

Amar asks you a series of questions:

What previous movies you watched? Which of these movies you liked or disliked? Is X a romantic movie? Does Salman Khan act in movie X?

….. And several other questions. Finally Amar tells you either Yes or No.

Amar is a **decision tree** for your movie preference.

But Amar may not give you accurate answer. In order to be more sure, you ask couple of friends – Rita, Suneet and Megha, the same question. They will each vote for the movie X. (This is **ensemble model aka Random Forest** in this case)

If you have a similar circle of friends, they may all have the exact same process of questions, so to avoid them all having the exact same answer, you’ll want to give them each a different sample from your list of movies. You decide to cut up your list and place it in a bag, then randomly draw from the bag, tell your friend whether or not you enjoyed that movie, and then place that sample back in the bag. This means you’ll be randomly drawing a sub sample from your original list with replacement (**bootstrapping** your original data)

And you will just make sure that , each of your friends asks different questions randomly.

Uses – Random Forest

Variable Selection: One of the best use cases for random forest is feature selection. One of the byproducts of trying lots of decision tree variations is that you can examine which variables are working best/worst in each tree.

Classification: Random forest is also great for classification. It can be used to make predictions for categories with multiple possible values