01.02.2024

BITS F464: Machine Learning

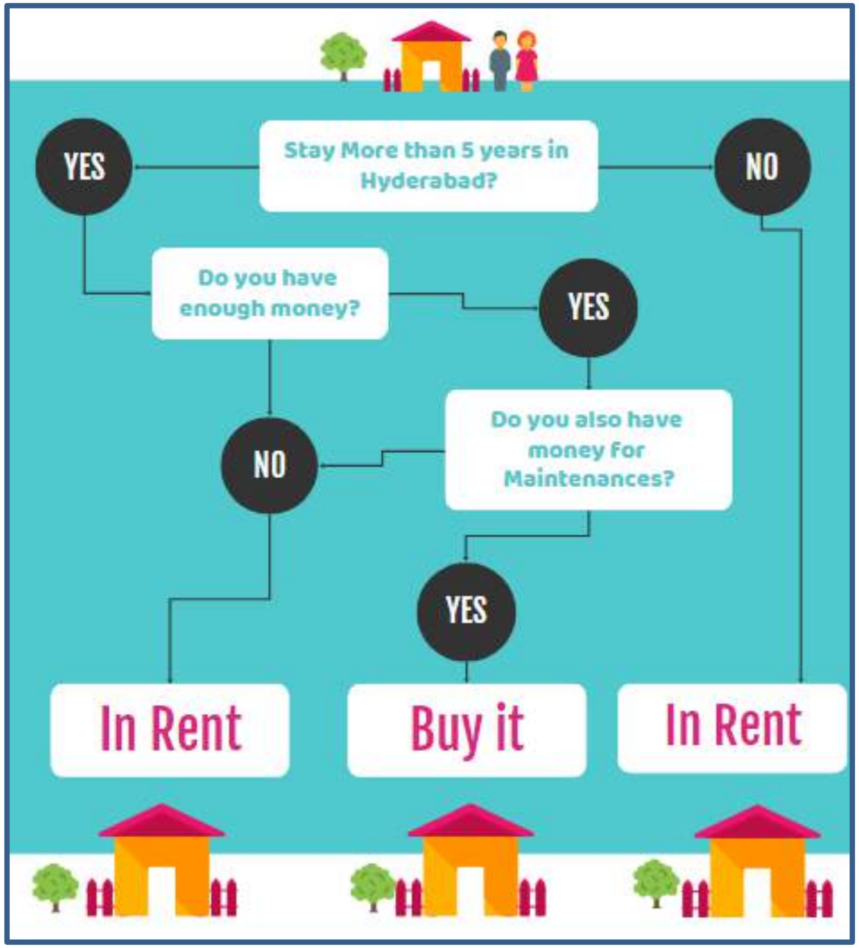
SYMBOLIC LEARNING: DECISION TREES

Chittaranjan Hota, Sr. Professor
Dept. of Computer Sc. and Information Systems
hota@hyderabad.bits-pilani.ac.in

Recap

- Concept Learning using Winston's Learning Program
- Overgeneralization Vs Overfitting
- Version space: Searching in Hypothesis space (Hill Climbing)
- Mitchell's Candidate Elimination algorithm
- Biased Hypothesis space (Bias in Learning)
- Inductive Bias in Version space
- Today, we will see Decision Trees (Another Inductive Learning algorithm under Symbolic ML)

Decision Tree: Applications



ENCODING

```
If (stay > 5yrs)
{
   If (money==50L)
   {
      If (m_money == 10L)
      {
         Buy a house;
       }
       Stay in rent;
   }
   Stay in rent;
}
Stay in rent;
```

- Equipment classification, Medical diagnosis, Credit Risk analysis, …

Applications continued…

XBox

Real-Time Human Pose Recognition in Parts from Single Depth Images

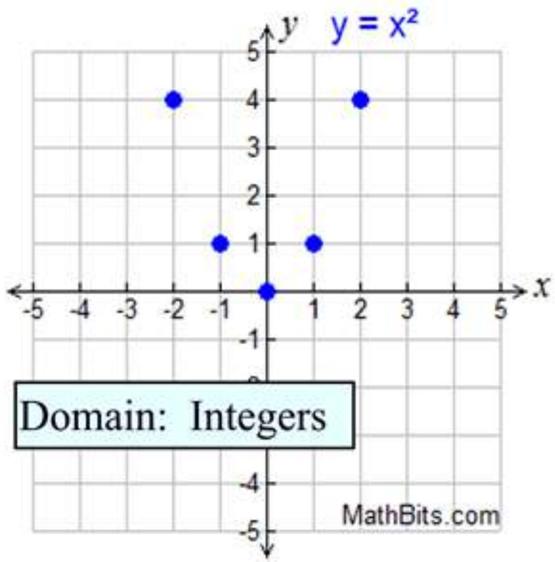
Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, Andrew Blake
Microsoft Research Cambridge & Xbox Incubation

HANSON ROBOTICS

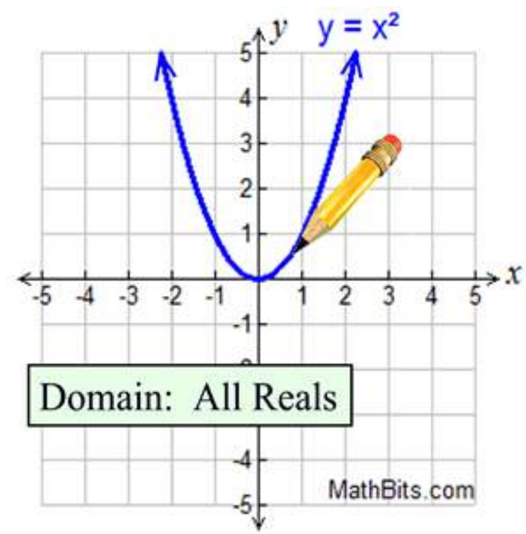
(Dialogues generated using Decision Trees)

What is Decision Tree Learning

- Decision tree learning is a method for approximating discrete-valued target functions, in which the learned function is represented by a decision tree.



- No. of students present.
- No. of holidays in this sem.
- No. of courses you finish.

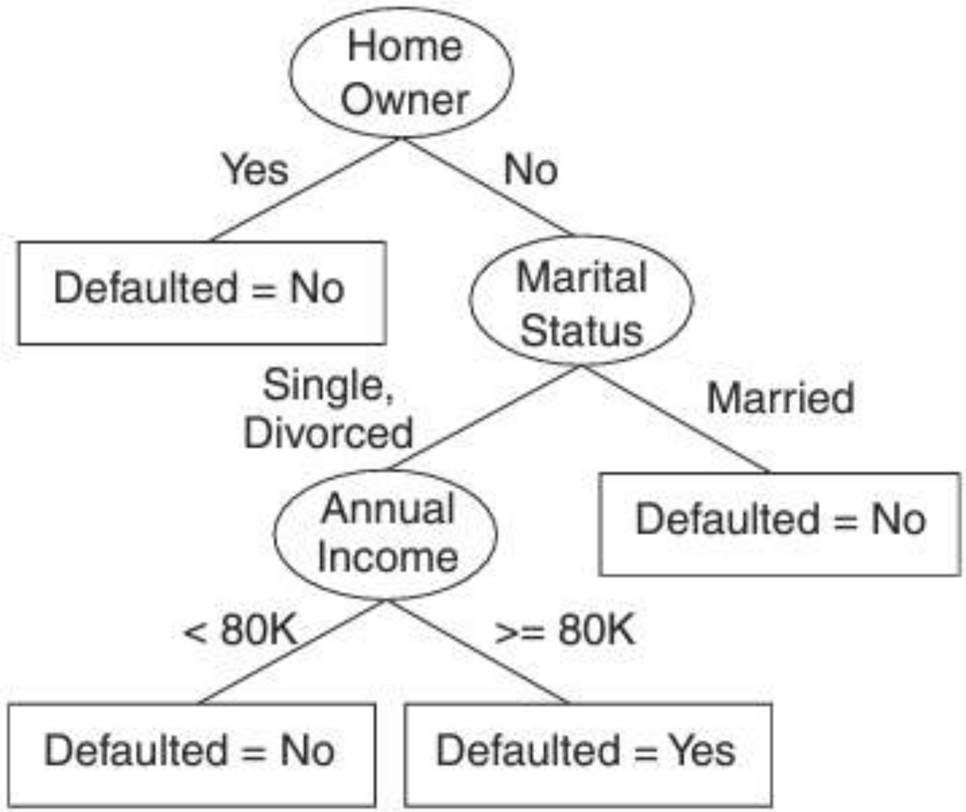


continuous

- Height of a person.
- Temperature in this room.
- USD value in rupees.

Decision Tree: Learning by Induction

	binary	categorical	continuous	class
Tid	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



(d)

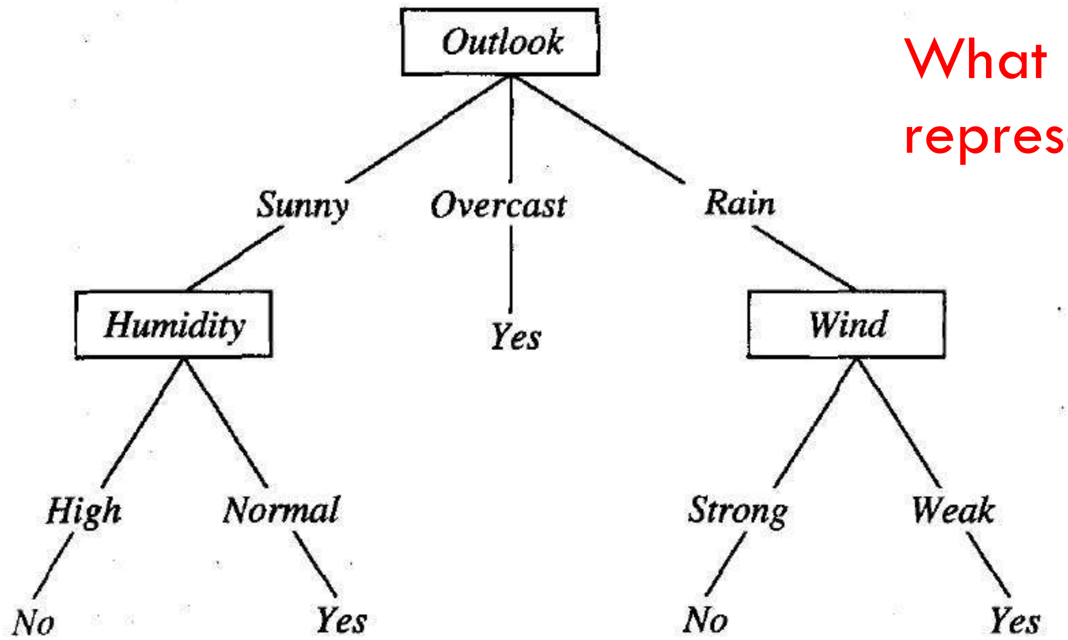
Continued...

- A decision tree is a non-parametric supervised learning algorithm, which is utilized for both classification and regression tasks.
- Classification: Playing Tennis, loan defaulter; Regression: How many students will enroll into ML next semester, What will be the cost of Honda Amaze next year?
- Parametric Vs Non-parametric models
    - A model learning from the data assuming a fixed number of parameters Vs. No-assumption or no prior-knowledge about data distribution (free to learn any functional form from data).
    - Linear/Logistic regression (coefficients), perceptron (weights) Vs. SVM, DT, KNN, Complex NNs.
    - Fast, Simple and Less data Vs. Slower, Complex and More data.

Decision Tree Representation

- Hypothesis space is disjunction of conjunctions, while candidate-elimination (version space) algorithm could only accommodate what? Given a test input:

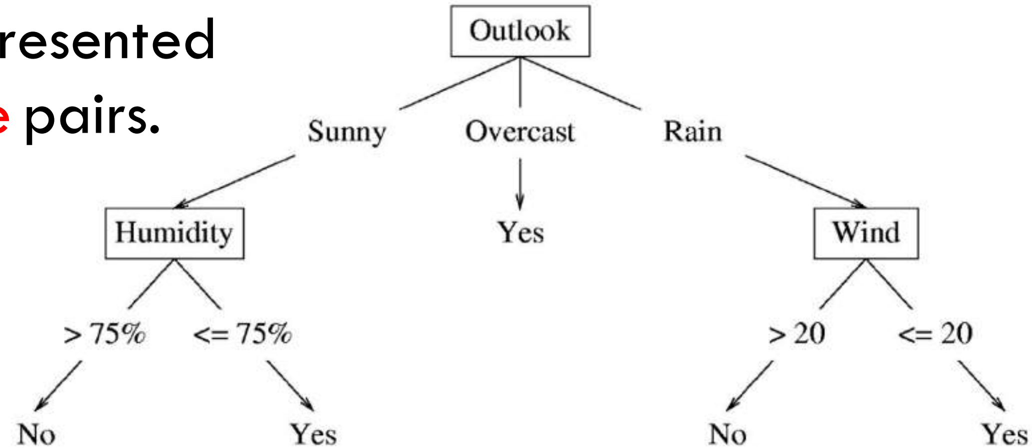
$$\langle Outlook = Sunny,\ Temperature = Hot,\ Humidity = High,\ Wind = Strong \rangle$$



What is the expression that represents this decision tree?

Problem Characteristics

- Instances are represented by attribute-value pairs.

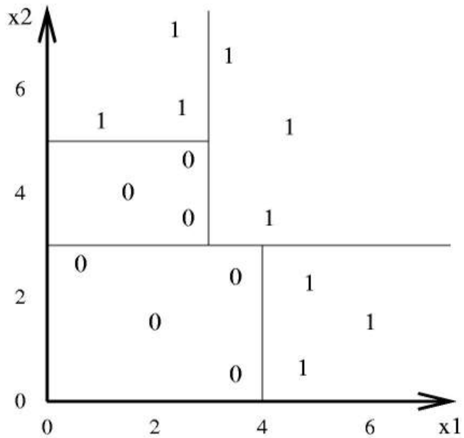


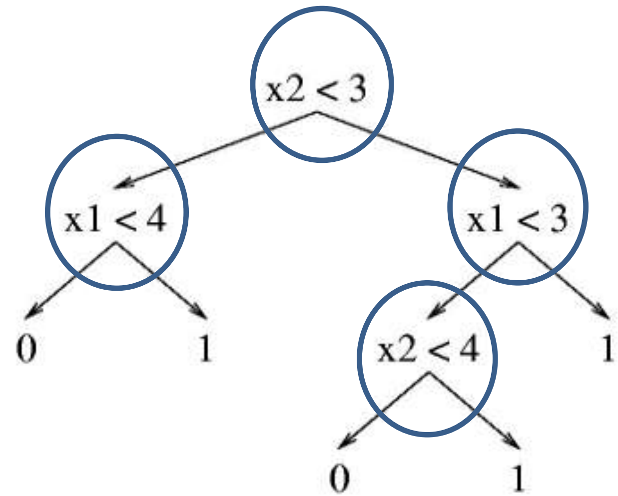
- In majority of the cases the target function has discrete output values. Either 2 or more output values are possible. However, real-valued outputs are also possible.
- Robust to errors: classification and attribute value errors.
- Training data may contain missing attribute values.

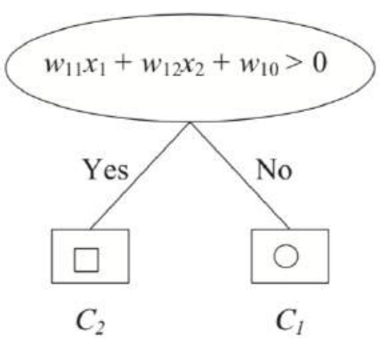
Decision Boundary in Decision Trees

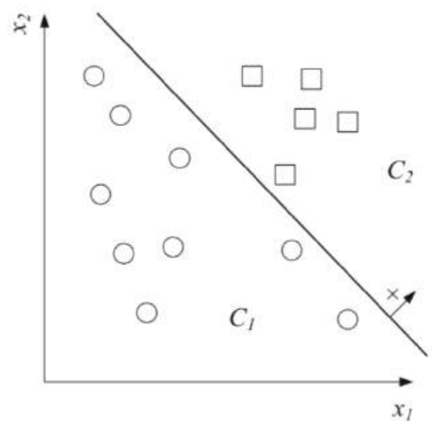
- Decision Trees divide the feature space into axis-parallel rectangles, and label each rectangle with one of the K-classes.

Univariate Tree:
In each internal node tree uses only one dimension or attribute. (ID3)









In a Multivariate tree, at a decision node, all input dimensions can be used and thus it is more general. (CART)

Decision Tree Learning Algorithm

Training set

Attribute set

TreeGrowth (E, F)

if stopping_cond$(E, F) = true$ **then**
  $leaf$ = createNode().
  $leaf.label$ = $\boxed{\texttt{Classify}(E)}$.
  return $leaf$.
else
  $root$ = createNode().
  $root.test\_cond$ = find_best_split(E, F).
  let $V = \{v \mid v$ is a possible outcome of $root.test\_cond\}$.
  for each $v \in V$ **do**
    $E_v = \{e \mid root.test\_cond(e) = v$ and $e \in E\}$.
    $child$ = TreeGrowth(E_v, F).
    add $child$ as descendent of $root$ and label the edge $(root \rightarrow child)$ as v.
  end for
end if
return $root$.

stopping_cond(): all the records have either the same class label or the same attribute values.

$$leaf.label = \underset{i}{\operatorname{argmax}}\ p(i|t)$$

Entropy, Gini Index, χ^2 statistic, …

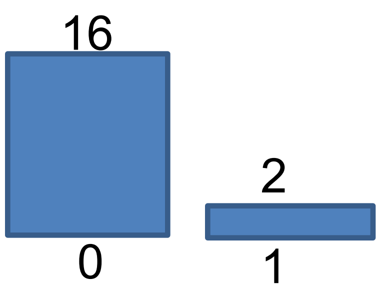
Quantifying Uncertainty (Info. Theoretic)

Entropy is a measure of the randomness/ uncertainty of the information being processed. Molecular disorder (Thermodynamics) to Shannon's Info Theory.

$$\mathrm{H}(X) := -\sum_{x\in\mathcal{X}} p(x)\log p(x)$$

Sequence 1:

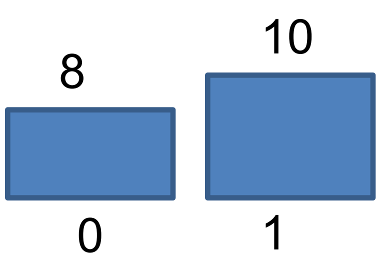
0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0



$$-\frac{16}{18}\log_2\frac{16}{18} - \frac{2}{18}\log_2\frac{2}{18} = 0.503$$

Sequence 2:

1 0 0 1 0 1 1 1 0 1 0 1 1 0 0 1 0 1

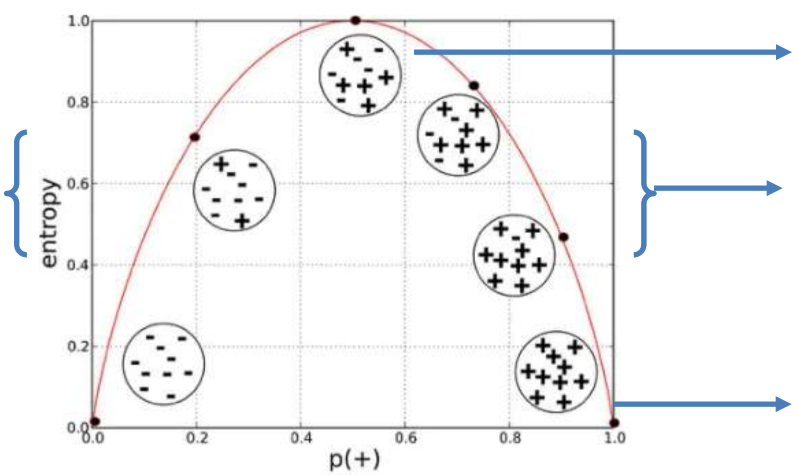


$$-\frac{8}{18}\log_2\frac{8}{18} - \frac{10}{18}\log_2\frac{10}{18} = 0.991$$

Optimal length code assigns $-\log_2 p$ bits to message having probability p.

Entropy Continued…

- Deterministic: good(all are true or false; one class in the leaf)
- Uniform distribution: bad (all classes in leaf equally probable)
- What about distributions in between?
- Entropy in information theory specifies the minimum number of bits needed to encode the class code of an instance.



Equal no. of + and - examples, what is H?

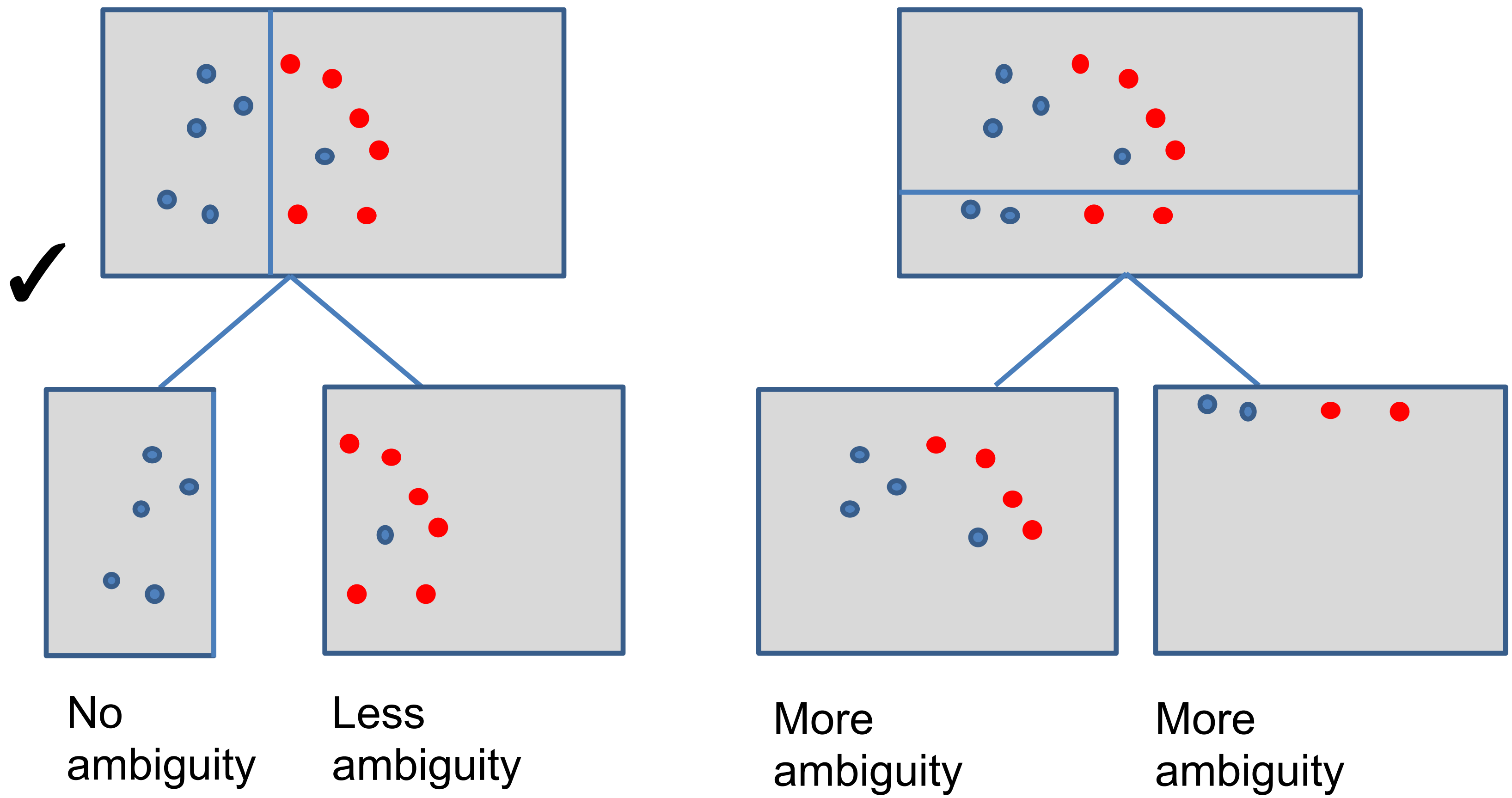
Unequal no. of +ve and -ve, what is H?

All members of S belong to the same class, what is H?

(Let S be a collection)

Choosing Attribute/value at each level

Which one is better?



Information Gain

- Measures how well an attribute divides the training examples according to their target types.
- Decline in Entropy.

divides

ID-3: Iterative Dichotomiser 3

$$Information\ Gain\ (S, A) = Entropy\ (S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

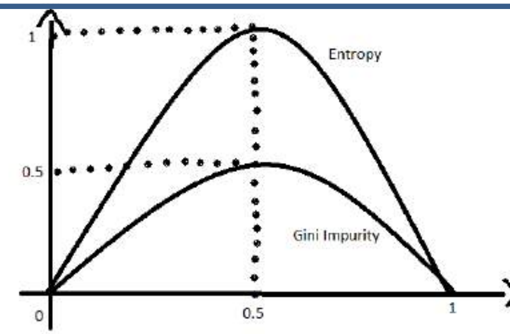
- Top down greedy heuristics: Ross Quinlan

$$\text{Gini Index} = 1 - \boxed{\sum_{i=1}^{n} (P_i)^2}$$

Gini purity

(Assignment 2)

Gini Impurity:



- Is a metric to measure how often a randomly chosen element would be incorrectly identified. Attribute with lower G.I should be preferred. CART

An Example: Play Tennis

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Entropy of the Training Set: $E(S) = E([9+,5-]) = (-9/14 \log_2 9/14) + (-5/14 \log_2 5/14) = 0.94$

Continued...

The information gain for Outlook is:

- I.G(S, Outlook) = E(S) – [5/14 * E(Outlook=sunny) + 4/14 * E(Outlook = overcast) + 5/14 * E(Outlook=rain)]
- I.G(S, Outlook) = E([9+,5-]) – [5/14 * E(2+,3-) + 4/14 * E([4+,0-]) + 5/14 * E([3+,2-])]
- I.G(S, Outlook) = 0.94 – [5/14 * 0.971 + 4/14 *0.0 + 5/14*0.971]
- I.G(S, Outlook) = 0.246

The information gain for Temperature is:

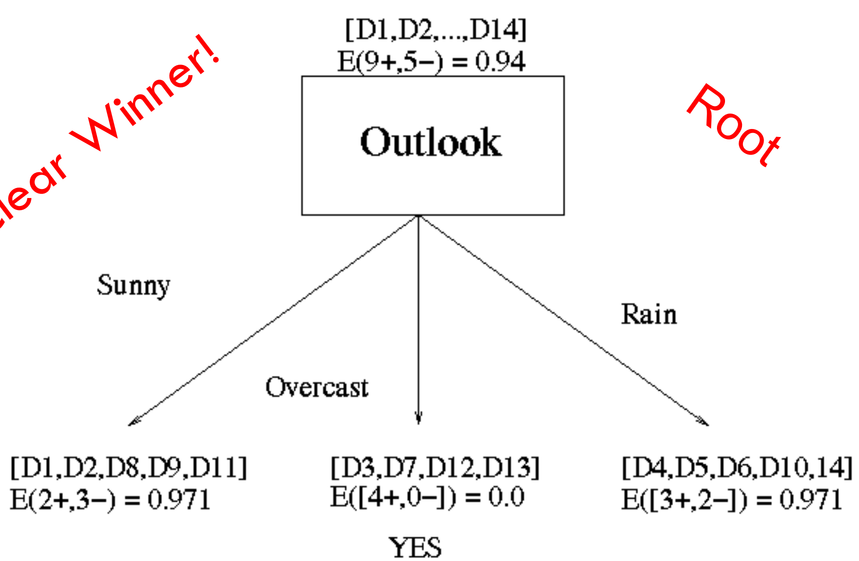
- I.G(S, Temperature) = 0.94 – [4/14 * E(Temperature=hot) + 6/14 * E(Temperature=mild) + 4/14 * E(Temperature=cool)]
- I.G(S, Temperature) = 0.94 – [4/14 * E([2+,2-]) + 6/14 * E([4+,2-]) + 4/14 * E([3+,1-])]
- I.G(S, Temperature) = 0.94 – [4/14 + 6/14*0.918 + 4/14*0.811]
- I.G(S, Temperature) = 0.029

Continued…

The information gain for Humidity is:

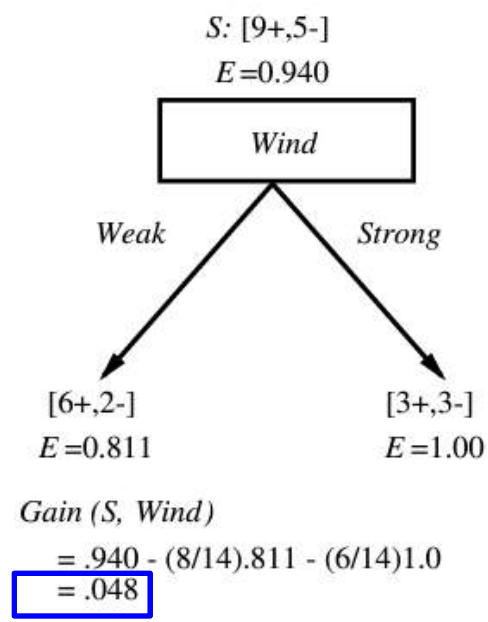
- I.G(S, Humidity) = 0.94 – [7/14 * E(Humidity=high) + 7/14 * E(Humidity=normal)]
- I.G(S, Humidity = 0.94 – [7/14 * E([3+,4-]) + 7/14 * E([6+,1-])]
- I.G(S, Humidity = 0.94 – [7/14 * 0.985 + 7/14 * 0.592]
- I.G(S, Humidity) = 0.1515

Clear Winner!



Root

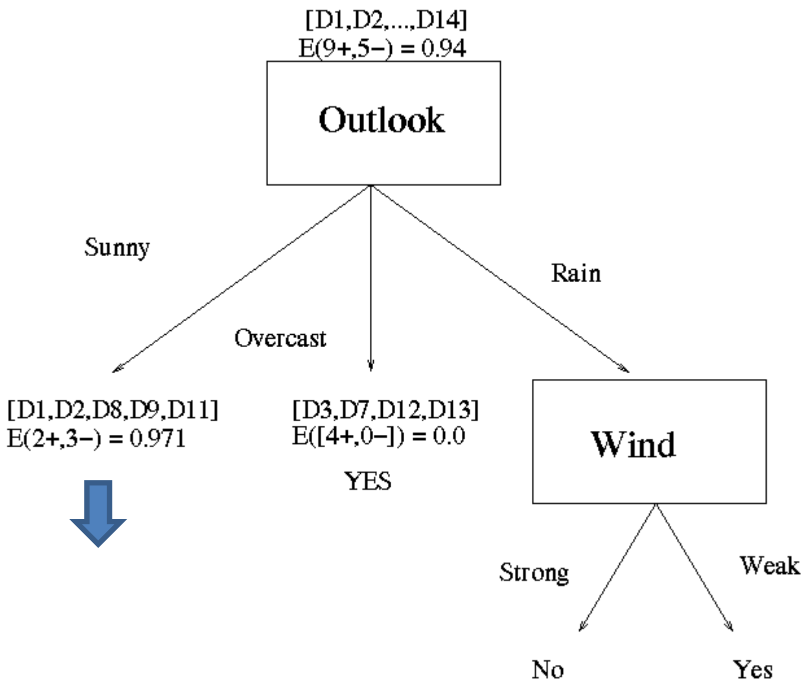
Information gain for Wind is:



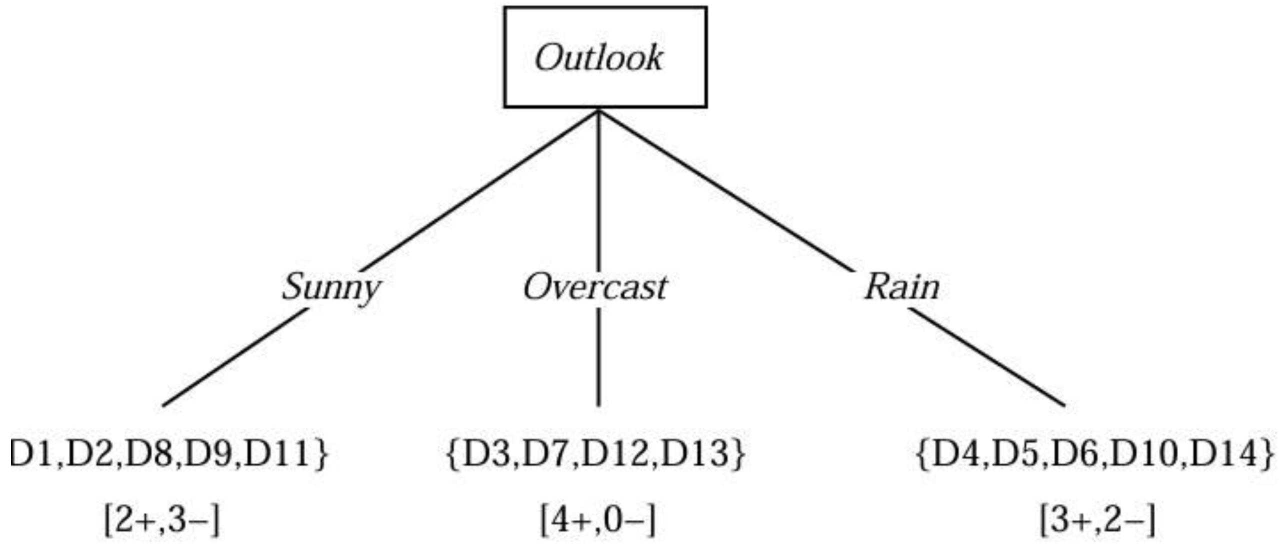
Continued…

We should find out the nodes that will go below Sunny, Overcast, and Rain:

- I.G(Outlook = Rain, Humidity) = 0.971 – [2/5 * E (Outlook = Rain $\wedge$ Humidity = high) + 3/5 * E (Outlook = Rain $\wedge$ Humidity = normal]
- I.G(Outlook = Rain, Humidity) = 0.02
- I.G(Outlook = Rain, Wind) = 0.971- [3/5*0 + 2/5*0]
- I.G(Outlook = Rain, Wind) = 0.971



Continued…



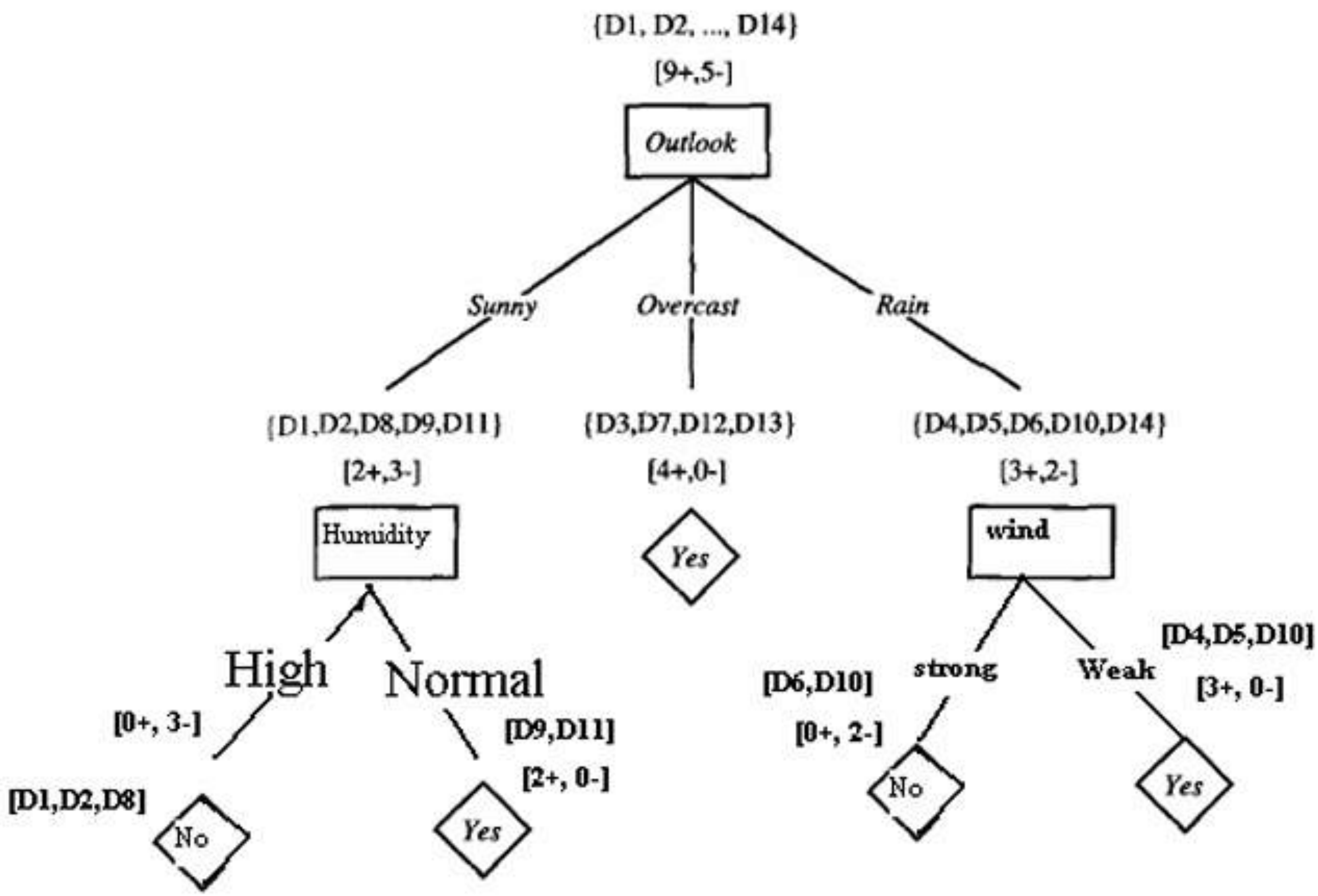
Yes

unny , *Humidity)* $= .970 - (3/5)\ 0.0 - (2/5)\ 0.0 = \boxed{.970}$

nny , *Temperature)* $= .970 - (2/5)\ 0.0 - (2/5)\ 1.0 - (1/5)\ 0.0 = .570$

unny , *Wind)* $= .970 - (2/5)\ 1.0 - (3/5)\ .918 = .019$

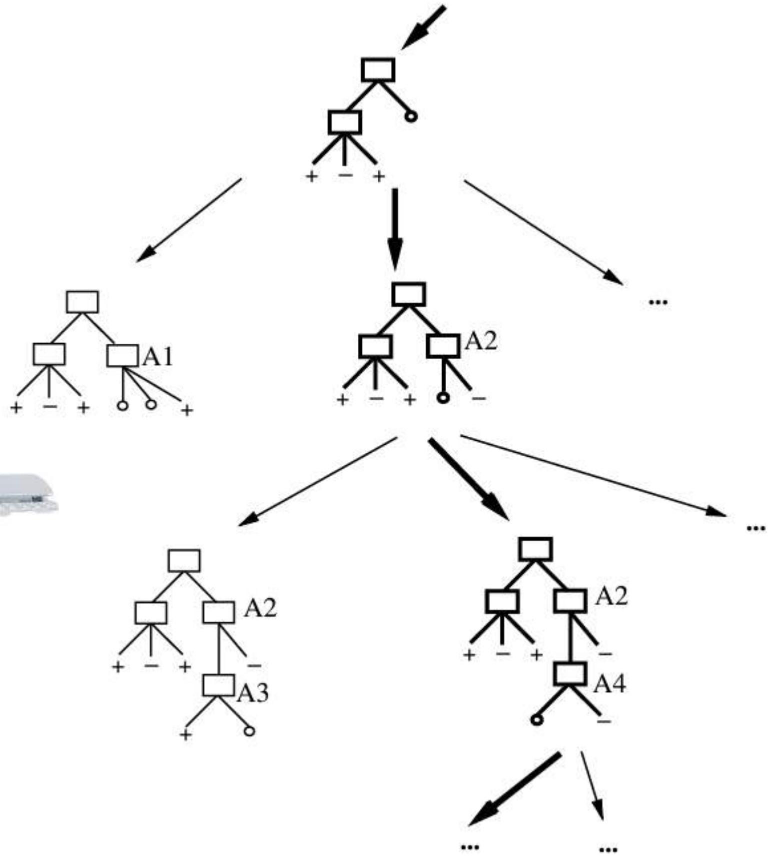
Continued…



{D1, D2, ..., D14}
[9+,5-]
Outlook
Sunny
Overcast
Rain
{D1,D2,D8,D9,D11}
[2+,3-]
{D3,D7,D12,D13}
[4+,0-]
{D4,D5,D6,D10,D14}
[3+,2-]
Humidity
Yes
wind
High
Normal
[0+, 3-]
[D9,D11]
[2+, 0-]
[D1,D2,D8]
No
Yes
[D6,D10]
[0+, 2-]
strong
Weak
[D4,D5,D10]
[3+, 0-]
No
Yes

Hypothesis space search by ID3

- Selects trees that place the attributes with highest information gain closest to the root.
- Selects in favor of shorter trees over longer ones.
- A smaller (simpler) tree is more general than a larger (complex) tree.
- Hence, smaller one will be more accurate.
- This is what Occam's Razor says, i.e.



(Inductive Bias)

When two hypothesis equally explain the training set, pick the more general of the two.

ID-3, C4.5 and CART

C4.5	Gain Ratio	Categorical and Numeric values	Pruning is used	Handles missing values.	Ross Quinlan
ID3	Information Gain	Only Categorical value	No pruning	No	Ross Quinlan
CART	Gini Index	Categorical and Numeric values	Pruning is used	Handles missing values.	Leo Breiman in 1984

Attributes with many values

- Gain Ratio attempts to lessen the bias of I.G on highly branched trees by introducing a normalizing term called the Intrinsic Information (II). The Intrinsic Information is defined as the entropy of sub-dataset proportions.

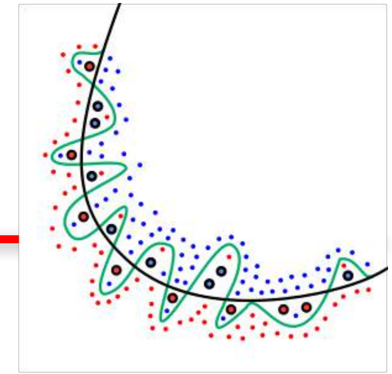
One approach: use $GainRatio$ instead

$$GainRatio(S, A) \equiv \frac{Gain(S, A)}{SplitInformation(S, A)}$$

$$SplitInformation(S, A) \equiv - \sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

where S_i is subset of S for which A has value v_i

Overfitting



Target function is : $Y = X_1 \wedge X_2$

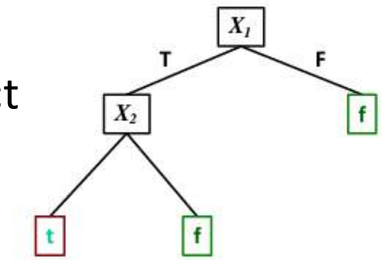
There is noise in some feature values.

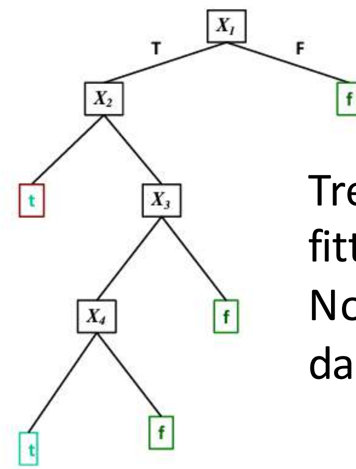
Training set:

X_1	X_2	X_3	X_4	X_5	...	Y
t	t	t	t	t	...	t
t	t	f	f	t	...	t
t	f	t	t	f	...	t
t	f	f	t	f	...	f
t	f	t	f	f	...	f
f	t	t	f	t	...	f

noisy value

Correct Tree

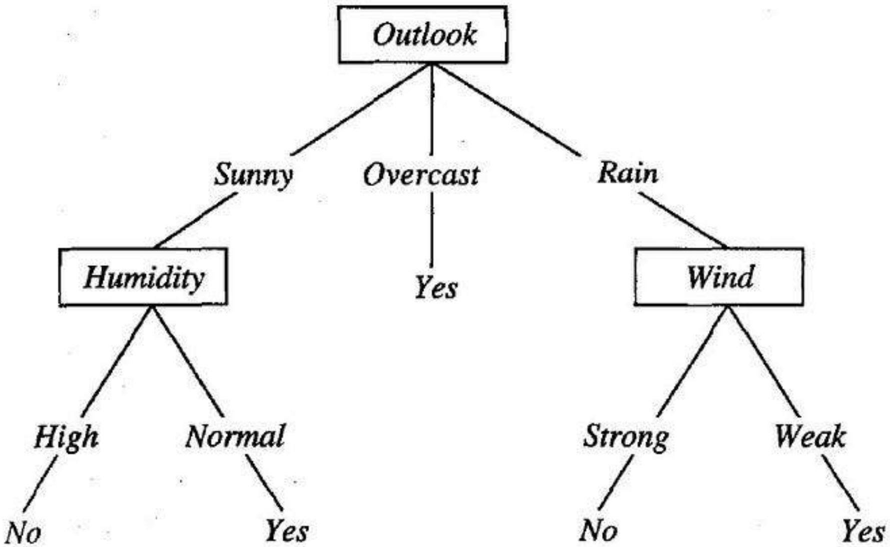




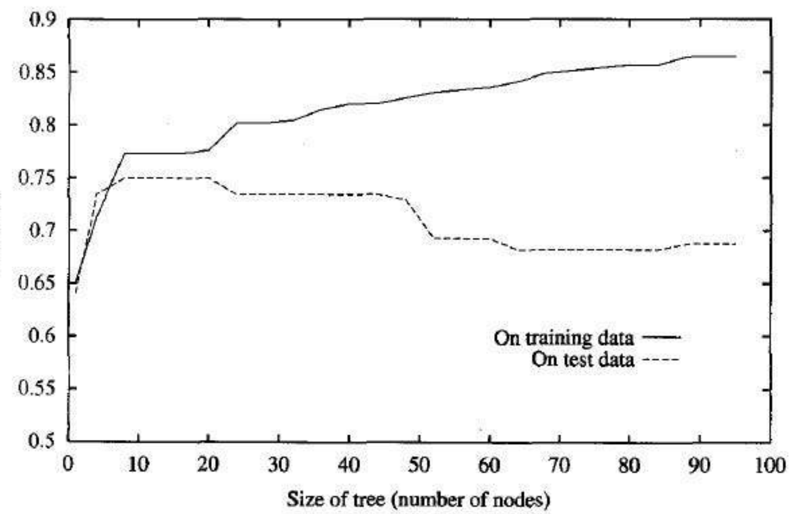
Tree fitting Noisy data

Overfitting in Decision Trees

- If the noisy sample D15 < Sunny, Hot, Normal, Strong, No > is incorrectly added to the previous set D1 to D14 [noisy because it would have been otherwise Yes (+ve)].



D9(+), D11(+), D15(-)



Noisy data and Variance could also cause this.

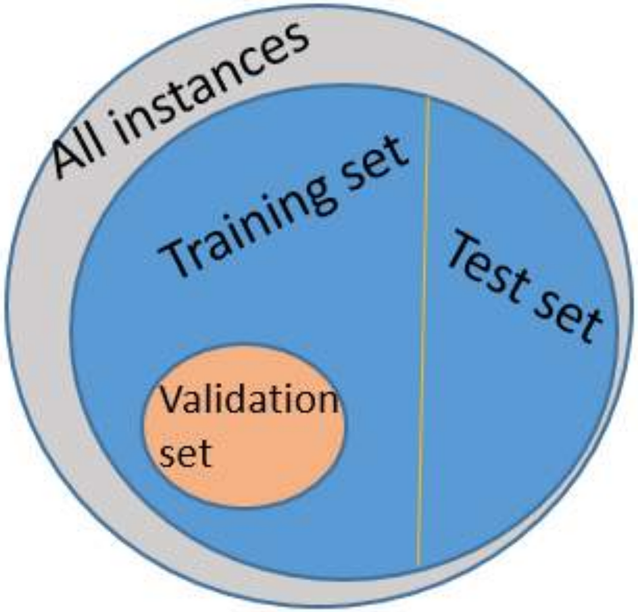
Avoiding Overfitting

How can we avoid overfitting?

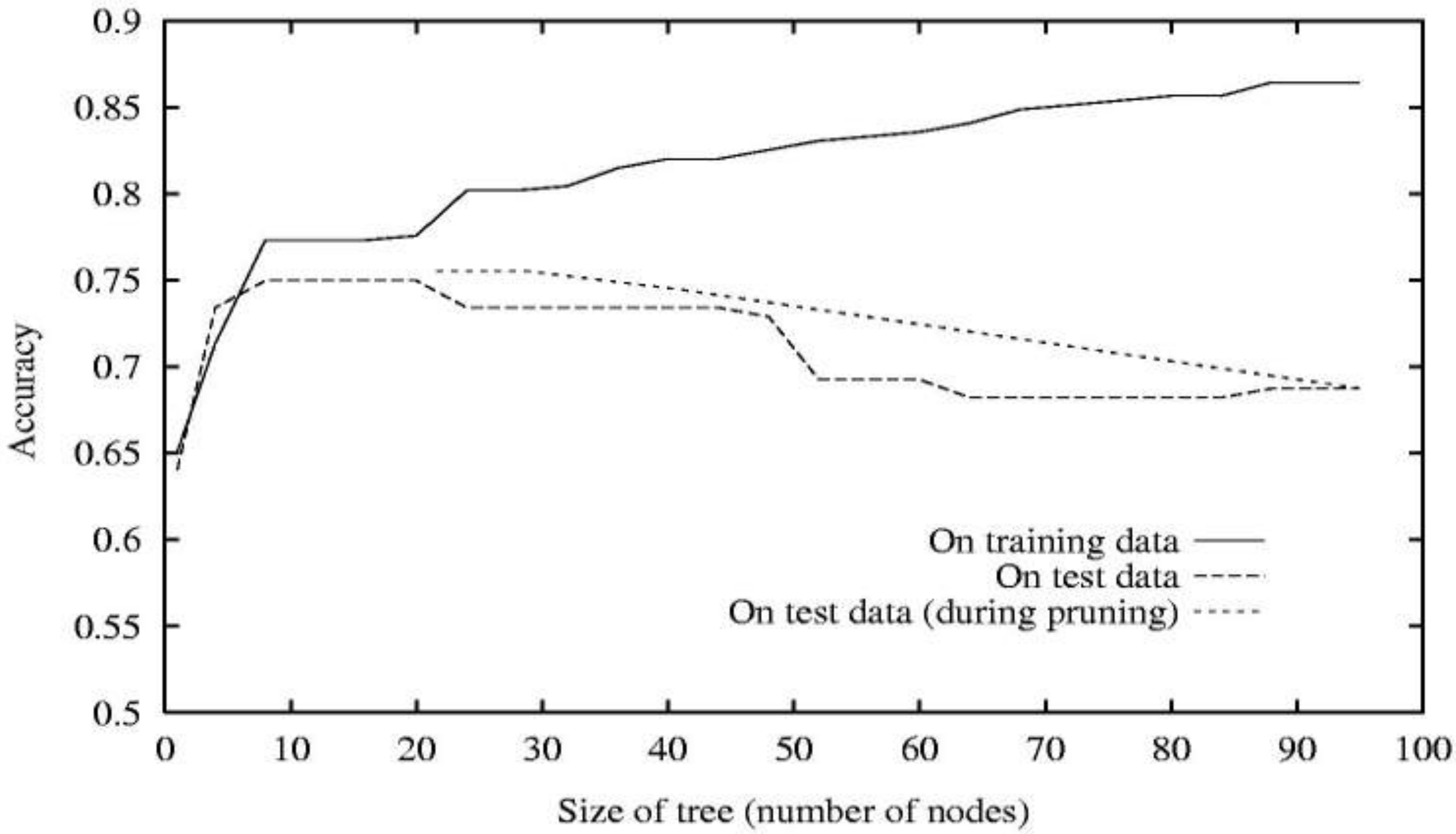
- Stop growing when data split :
  significant
- Grow full tree, then post-prun

How to select "best" tree:

- Measure performance over trai
- Measure performance over separate validation data set
- Add complexity penalty to performance measure



Reduced Error Pruning

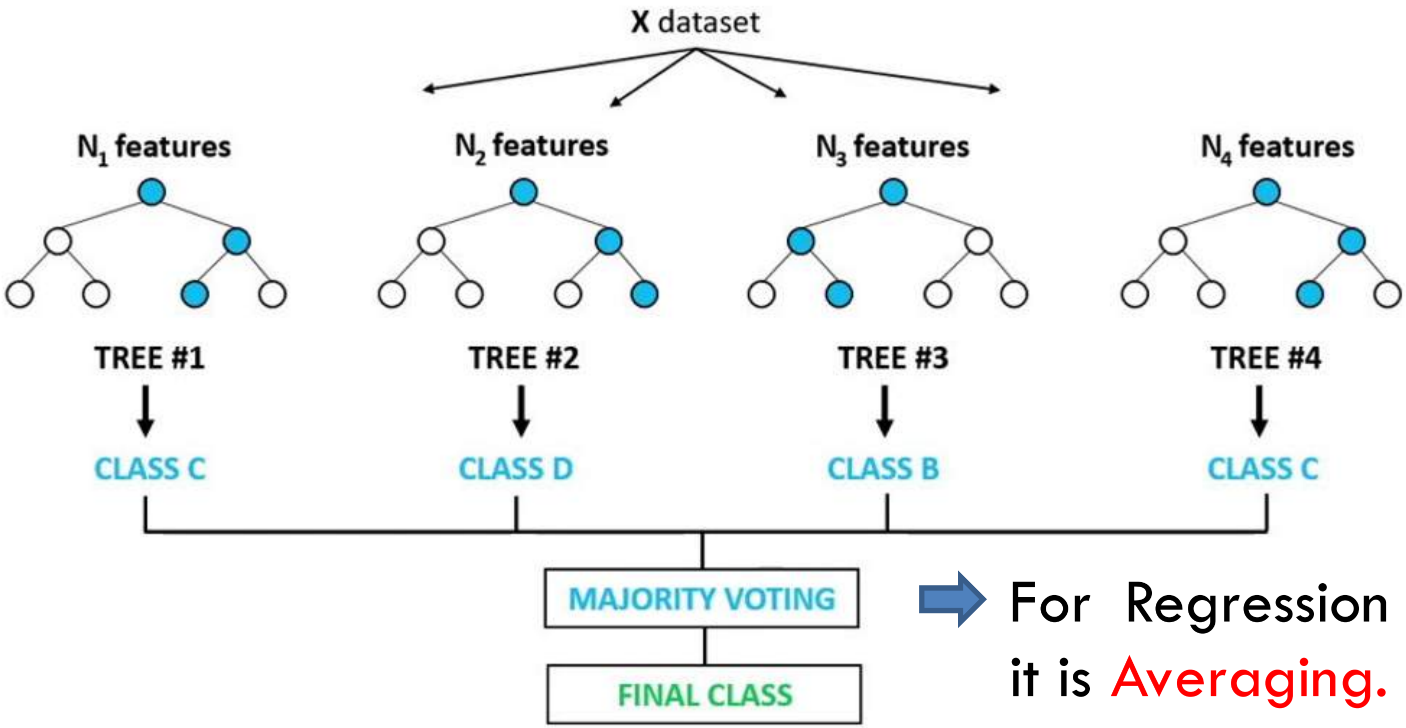


Rule Post Pruning

1. Convert tree to equivalent set of rules
2. Prune each rule independently of others
3. Sort final rules into desired sequence for use

Random Forest (Decision Forest)

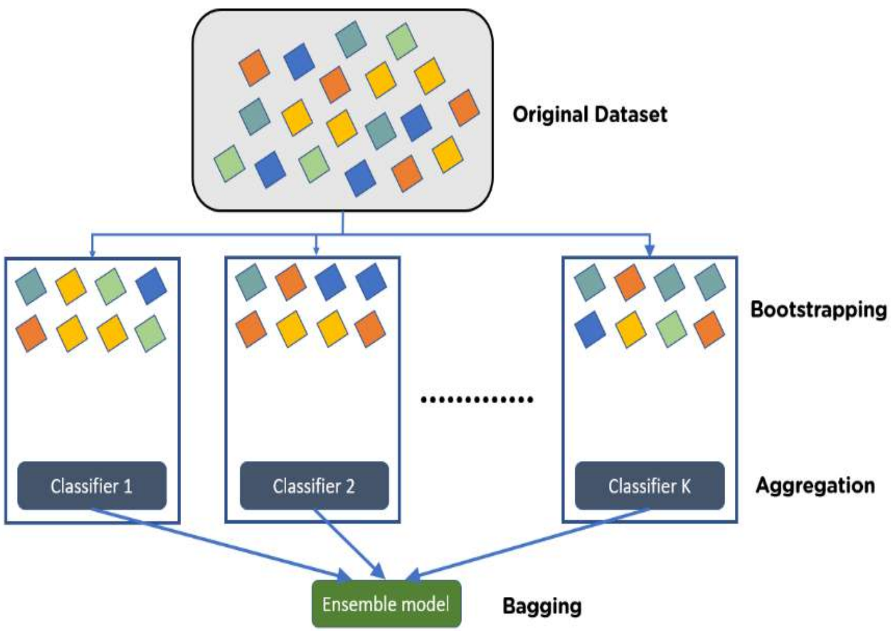
- Another way to avoid Bias and Overfitting in Decision Trees.
- It is an Ensemble (with bagging and boosting)

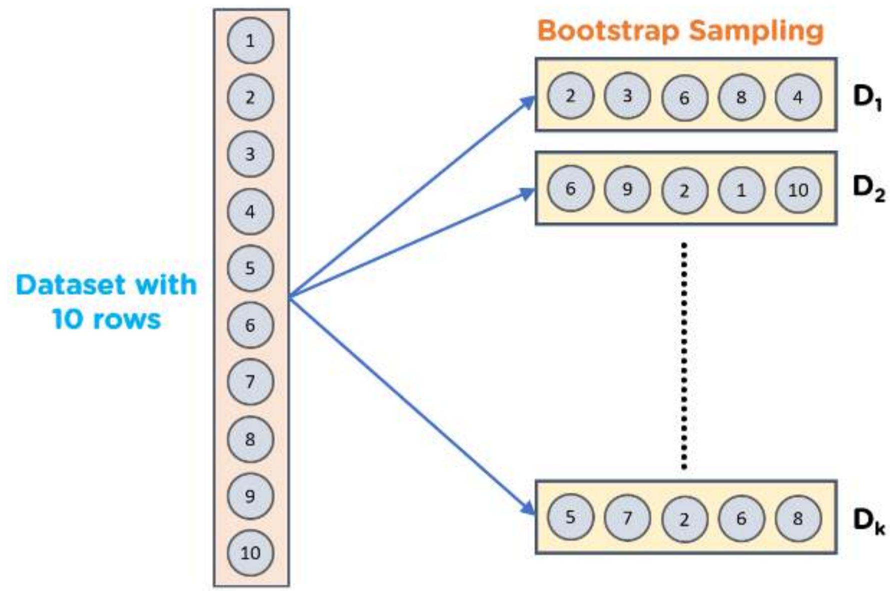


For Regression tasks, it is Averaging.

Image source: https://www.kaggle.com/

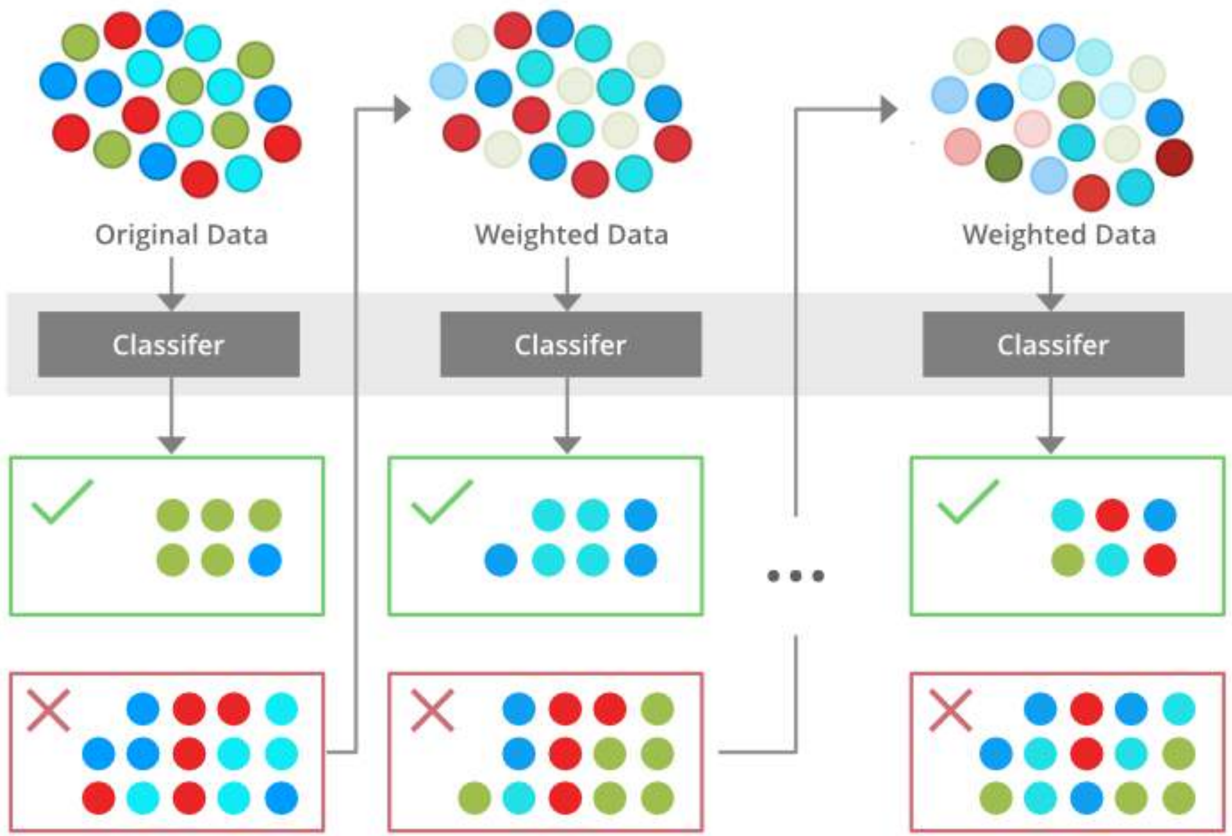
Bagging in Random Forest





Img. Source: https://www.geeksforgeeks.org/

Boosting in Gradient Boosted DTs



Img. Source: https://www.geeksforgeeks.org/

TensorFlow's Decision Forest (Assignment 2)

```
!pip install tensorflow_decision_forests
# TF-DF requires Tensorflow < 2.15 or tf_keras
!pip install tf_keras
```

```
import os
# Keep using Keras 2
os.environ['TF_USE_LEGACY_KERAS'] = '1'

import tensorflow_decision_forests as tfdf

import numpy as np
import pandas as pd
import tensorflow as tf
import tf_keras
import math
```

```
[8]
# Load a dataset into a Pandas Dataframe.
# dataset_df = pd.read_excel("/content/Data-RF.xlsx")
dataset_df = pd.read_excel("/content/drive/MyDrive/Data-RF.xlsx")

# Display the first 3 examples.
dataset_df.head(3)
```

	Lab-Test1(30)	Lab-Test2(24)	Midsem Test (90)	Gender	Attendance	Grade
0	13.00	24	66.0	Male	High	A
1	15.00	24	67.0	Female	High	A
2	5.25	24	45.0	Male	High	B-

```
# Encode the categorical labels as integers.
#
# Details:
# This stage is necessary if your classification label is represented as a
# string since Keras expects integer classification labels.
# When using `pd_dataframe_to_tf_dataset` (see below), this step can be skipped.

# Name of the label column.
label = "Grade"

classes = dataset_df[label].unique().tolist()
print(f"Label classes: {classes}")

dataset_df[label] = dataset_df[label].map(classes.index)
```

```
Label classes: [0, 1, 2, 3, 4, 5, 6, 7]
```

```
# Split the dataset into a training and a testing dataset.

def split_dataset(dataset, test_ratio=0.20):
  """Splits a panda dataframe in two."""
  test_indices = np.random.rand(len(dataset)) < test_ratio
  return dataset[~test_indices], dataset[test_indices]


train_ds_pd, test_ds_pd = split_dataset(dataset_df)
print("{} examples in training, {} examples for testing.".format(
    len(train_ds_pd), len(test_ds_pd)))
```

```
399 examples in training, 101 examples for testing.
```

Training accuracy for 10 DTs (CART)

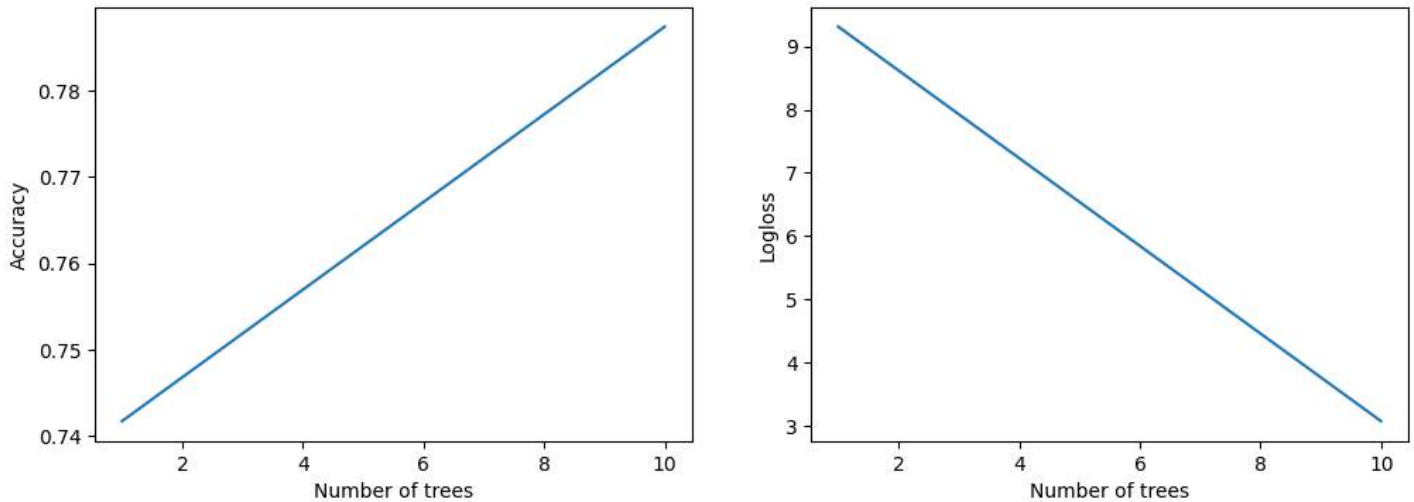
```
# https://www.tensorflow.org/decision_forests/api_docs/python/tfdf/keras/RandomForestModel
# Specify the model.
model_1 = tfdf.keras.RandomForestModel(verbose=2, categorical_algorithm="CART", num_trees=10, max_depth=16)

# Train the model.
model_1.fit(train_ds)
```

```
  compute_oob_performances: true
  compute_oob_variable_importances: false
  num_oob_variable_importances_permutations: 1
  bootstrap_training_dataset: true
  bootstrap_size_ratio: 1
  adapt_bootstrap_size_ratio_for_maximum_training_duration: false
  sampling_with_replacement: true
}

[INFO 24-01-31 12:19:25.8820 UTC kernel.cc:825] Deployment config:
cache_path: "/tmp/tmpnlyusd4l/working_cache"
num_threads: 2
try_resume_training: true

[INFO 24-01-31 12:19:25.8831 UTC kernel.cc:887] Train model
[INFO 24-01-31 12:19:25.8833 UTC random_forest.cc:416] Training random forest on 399 example(s) and 5 feature(s).
[INFO 24-01-31 12:19:25.8851 UTC random_forest.cc:802] Training of tree  1/10 (tree index:0) done accuracy:0.73125 logloss:9.68673
[INFO 24-01-31 12:19:25.9022 UTC random_forest.cc:802] Training of tree  10/10 (tree index:5) done accuracy:0.787342 logloss:3.07538
[INFO 24-01-31 12:19:25.9025 UTC random_forest.cc:882] Final OOB metrics: accuracy:0.787342 logloss:3.07538
```

Training accuracy for 10 trees (Plot)

```python
import matplotlib.pyplot as plt

logs = model_1.make_inspector().training_logs()

plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)
plt.plot([log.num_trees for log in logs], [log.evaluation.accuracy for log in logs])
plt.xlabel("Number of trees")
plt.ylabel("Accuracy")

plt.subplot(1, 2, 2)
plt.plot([log.num_trees for log in logs], [log.evaluation.loss for log in logs])
plt.xlabel("Number of trees")
plt.ylabel("Logloss")

plt.show()
```

Log loss, also known as logarithmic loss or cross-entropy loss, is a common evaluation metric that quantifies the difference between predicted probabilities and actual values.



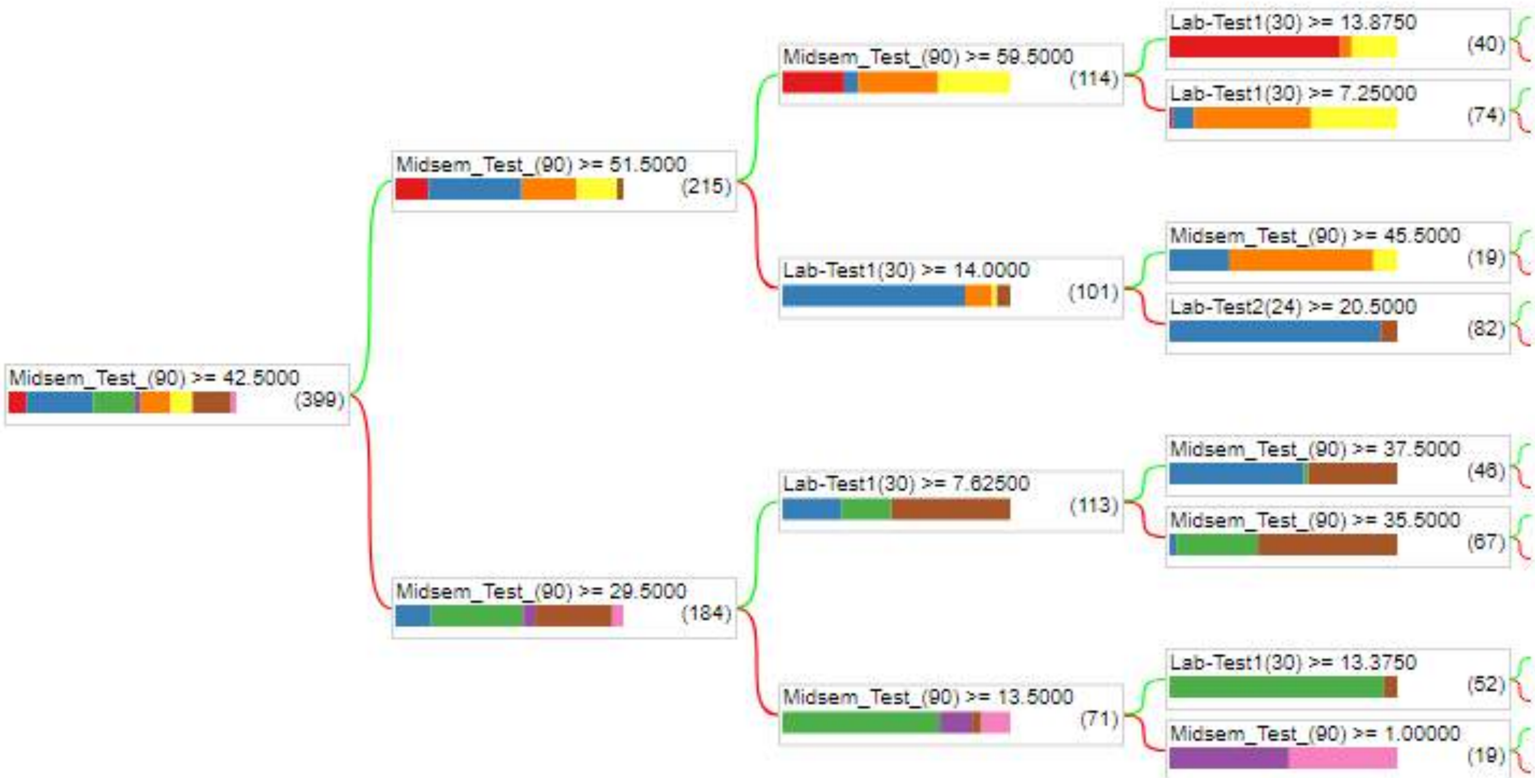
Testing accuracy for 10 DTs (CART)

```
[14] model_1.compile(metrics=["accuracy"])
     evaluation = model_1.evaluate(test_ds, return_dict=True)
     print()

     for name, value in evaluation.items():
       print(f"{name}: {value:.8f}")
```

```
1/1 [==============================] - 5s 5s/step - loss: 0.0000e+00 - accuracy: 0.8515

loss: 0.00000000
accuracy: 0.85148513
```

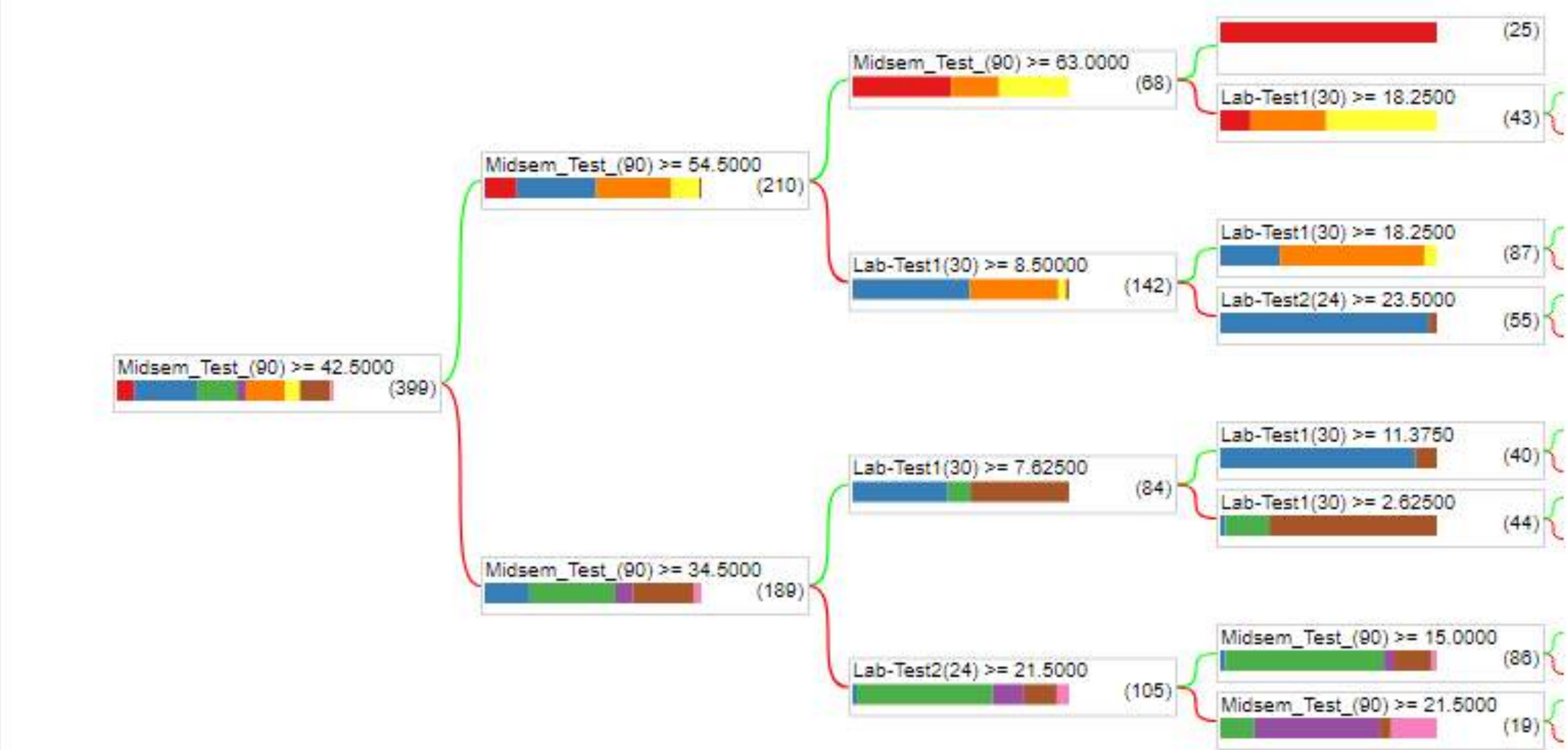
Plot of Individual DTs (index 0)

```
[15] tfdf.model_plotter.plot_model_in_colab(model_1, tree_idx=0, max_depth=3)
```



Plot of Individual DTs (index 1)

```
tfdf.model_plotter.plot_model_in_colab(model_1, tree_idx=1, max_depth=3)
```



Training accuracy for 30 DTs

```
[33] # https://www.tensorflow.org/decision_forests/api_docs/python/tfdf/keras/RandomForestModel
     # Specify the model.
     model_1 = tfdf.keras.RandomForestModel(verbose=2, categorical_algorithm="CART", num_trees=30, max_depth=16)

     # Train the model.
     model_1.fit(train_ds)
```

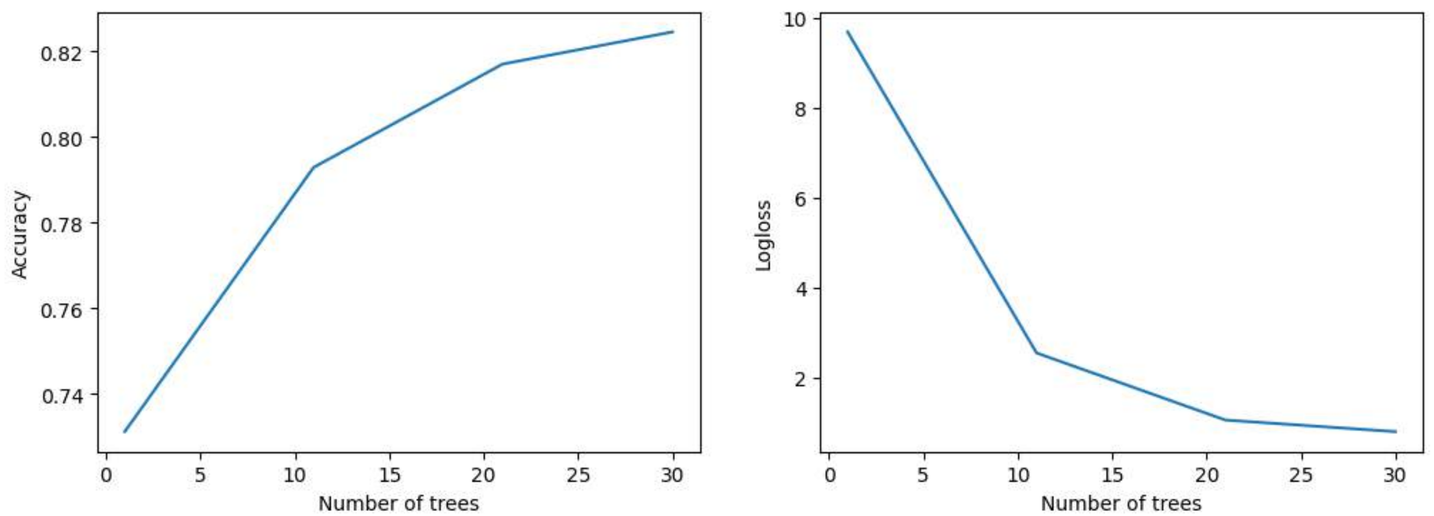
```
  num_oob_variable_importances_permutations: 1
  bootstrap_training_dataset: true
  bootstrap_size_ratio: 1
  adapt_bootstrap_size_ratio_for_maximum_training_duration: false
  sampling_with_replacement: true
}

[INFO 24-01-31 12:14:46.9634 UTC kernel.cc:825] Deployment config:
cache_path: "/tmp/tmpng4jttqv/working_cache"
num_threads: 2
try_resume_training: true

[INFO 24-01-31 12:14:46.9637 UTC kernel.cc:887] Train model
[INFO 24-01-31 12:14:46.9639 UTC random_forest.cc:416] Training random forest on 399 example(s) and 5 feature(s).
[INFO 24-01-31 12:14:46.9653 UTC random_forest.cc:802] Training of tree  1/30 (tree index:0) done accuracy:0.73125 logloss:9.68673
[INFO 24-01-31 12:14:46.9739 UTC random_forest.cc:802] Training of tree  11/30 (tree index:10) done accuracy:0.792929 logloss:2.54161
[INFO 24-01-31 12:14:46.9822 UTC random_forest.cc:802] Training of tree  21/30 (tree index:20) done accuracy:0.817043 logloss:1.0483
[INFO 24-01-31 12:14:46.9896 UTC random_forest.cc:802] Training of tree  30/30 (tree index:29) done accuracy:0.824561 logloss:0.790764
[INFO 24-01-31 12:14:46.9896 UTC random_forest.cc:882] Final OOB metrics: accuracy:0.824561 logloss:0.790764
```

Testing accuracy for 30 trees did not improve from that of 10 trees. (0.85148)

Training accuracy for 30 DTs (Plot)

```
import matplotlib.pyplot as plt

logs = model_1.make_inspector().training_logs()

plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)
plt.plot([log.num_trees for log in logs], [log.evaluation.accuracy for log in logs])
plt.xlabel("Number of trees")
plt.ylabel("Accuracy")

plt.subplot(1, 2, 2)
plt.plot([log.num_trees for log in logs], [log.evaluation.loss for log in logs])
plt.xlabel("Number of trees")
plt.ylabel("Logloss")

plt.show()
```

You will have to compare the accuracies with Gradient Boosted Decision Trees.



https://www.tensorflow.org/decision_forests/tutorials/beginner_colab

Thank you!