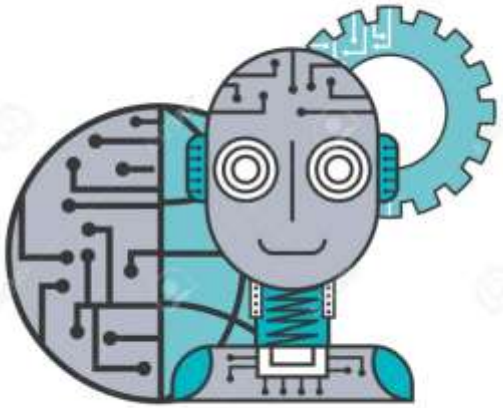
UNIVERSITY OF DERBY

Module 6CM521: Frontiers in AI and Data Science



Classification Problem and Decision Tree

Ioannis Tsioulis (MC)

TOP 30 GUARDIAN UNIVERSITY GUIDE 2019

Slides acknowledgements:
Dr Aaisha Makkar

Declaration

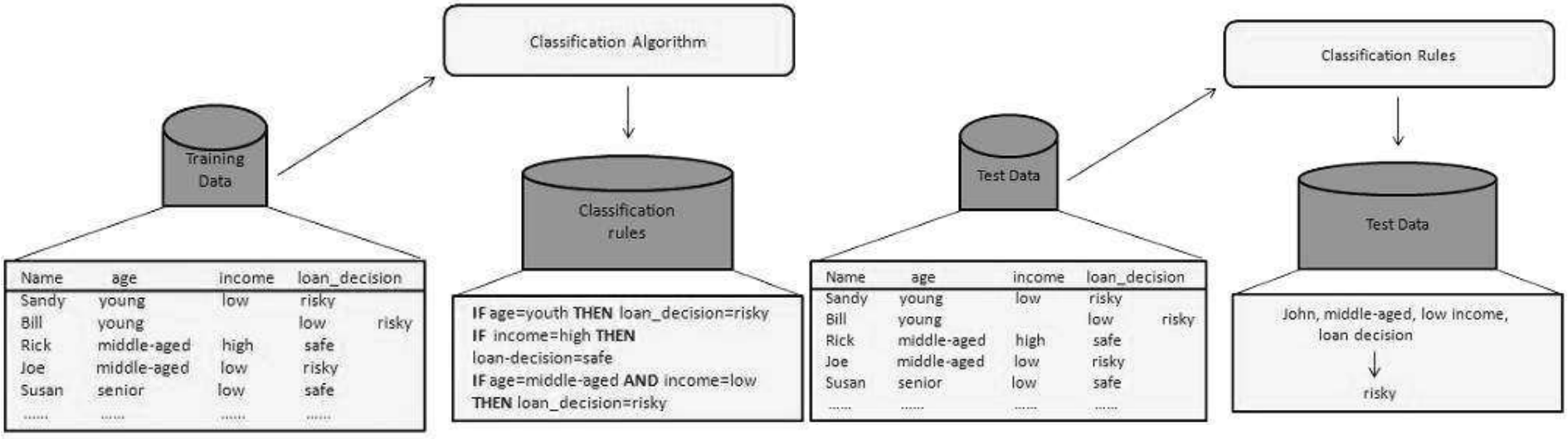
These slides are organised based on Chapter 8 in book Data Mining Concept and Techniques. Other online resources that help to create the slides are referenced at the end.

Acknowledgement

Most of the slides are adapted from the lecture slides of Dr Hong Qing Yu (Harry).

Classification problems in machine learning

- The task of classification is to use a machine learning algorithm that learn how to assign a class label to the training dataset and provide good performance on testing dataset from a specific problem domain.
- General approach:



UNIVERSITY OF DERBY

Different types of classification problems

From label size or property point of view, there are 4 different classification problems

- Binary classification
- Multi-class classification
- Multi-label classification
- Imbalanced classification

TOP 30 GUARDIAN UNIVERSITY GUIDE 2019

Binary Classification

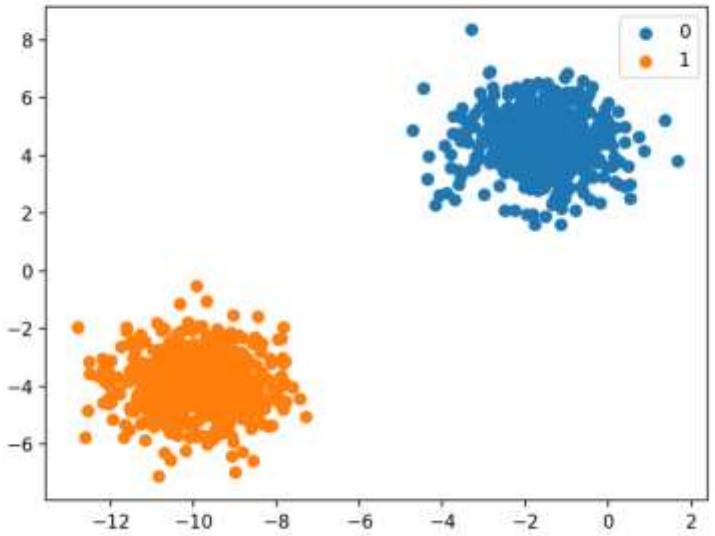
Binary classification refers to those classification tasks that have only two class labels. Typically, binary classification tasks involve one class that is the normal state and another class that is the abnormal state.

Examples include:

- Email spam detection (spam or not).
- Churn prediction (churn or not).
- Conversion prediction (buy or not).

Popular algorithms

- Logistic Regression
- k-Nearest Neighbours
- Decision Trees
- Support Vector Machine
- Naive Bayes



Multi-Class Classification

Multi-Class Classification refers to those classification tasks that have more than two class labels.

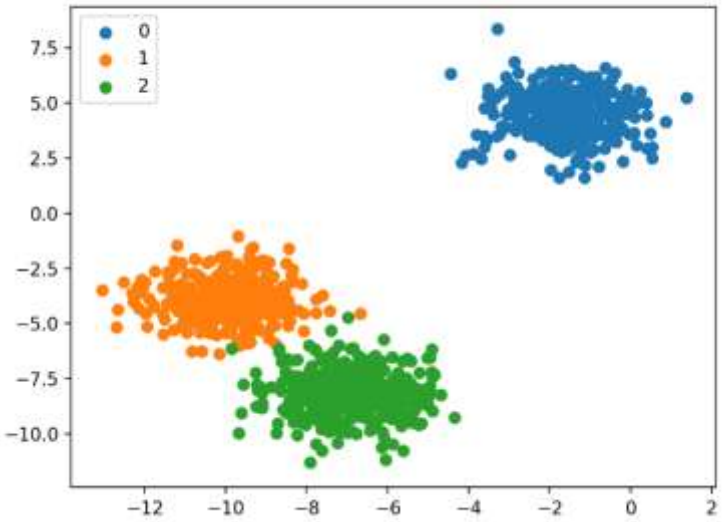
Examples include:

- Face classification.
- Plant species classification.
- Optical character recognition.



Popular algorithms:

- k-Nearest Neighbours.
- Decision Trees.
- Naive Bayes.
- Random Forest.
- Gradient Boosting.



TOP 30 GUARDIAN UNIVERSITY GUIDE 2019

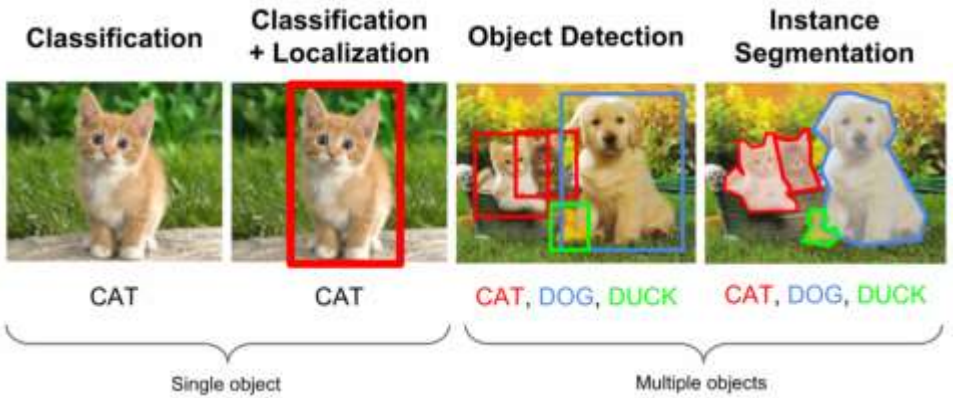
Multi-Label Classification

Multi-Label Classification refers to those classification tasks that have two or more class labels, where one or more class labels may be predicted for each example.

Consider the example of photo classification, where a given photo may have multiple objects in the scene and a model may predict the presence of multiple known objects in the photo, such as "*bicycle*," "*apple*," "*person*," etc.

Specialized versions of standard classification algorithms needs to be implemented, so-called multi-label versions of the algorithms, including:

- Multi-label Decision Trees
- Multi-label Random Forests
- Multi-label Gradient Boosting

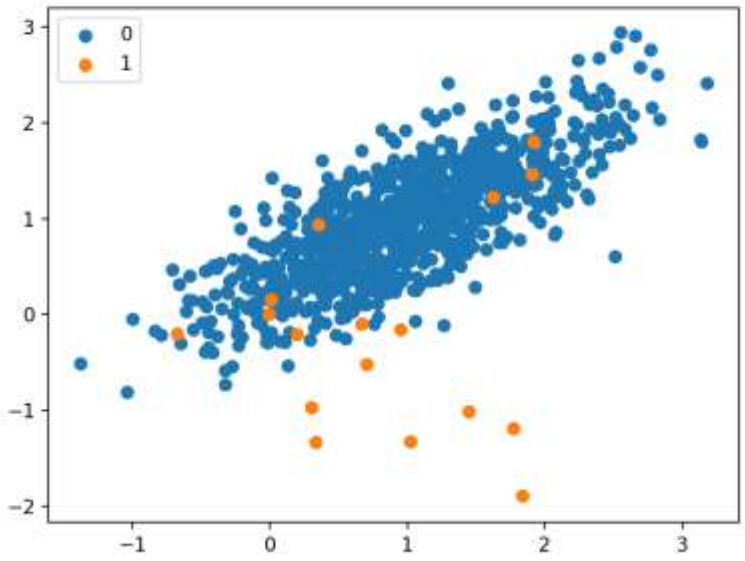


Imbalanced Classification

Imbalanced Classification refers to classification tasks where the number of examples in each class is unequally distributed.

Random Undersampling:

- **Random Oversampling**: Randomly duplicate examples in the minority class.
- **Random Undersampling**: Randomly delete examples in the majority class.



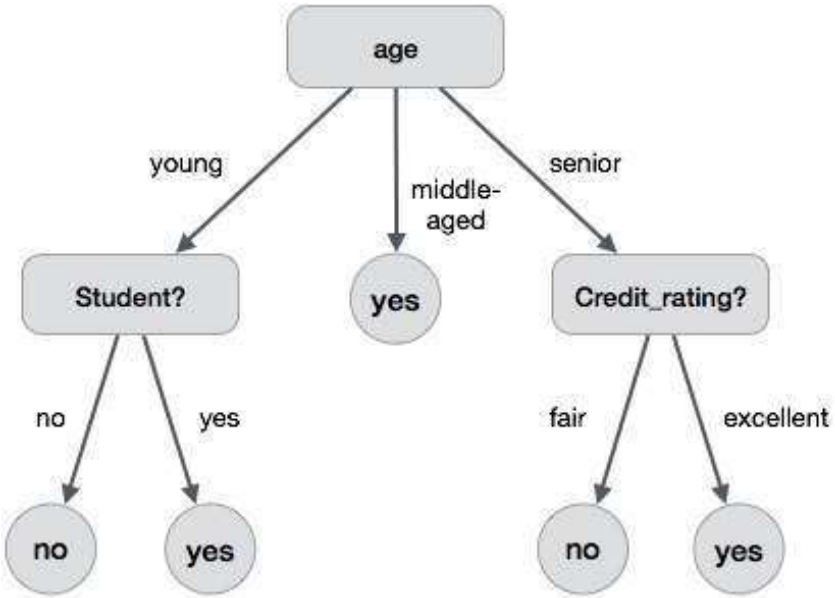
SMOTE Oversampling:
duplicating examples from the minority class

Decision Tree (Induction/Learning)

- Decision tree induction is one of the simplest, and yet most successful forms of learning algorithm for solving classification problem.
- It also serves as a good introduction to the area of inductive learning in AI domain.
- See example

The benefits:

- It does not require any domain knowledge.
- It is easy to comprehend.
- The learning and classification steps of a decision tree are simple and fast.

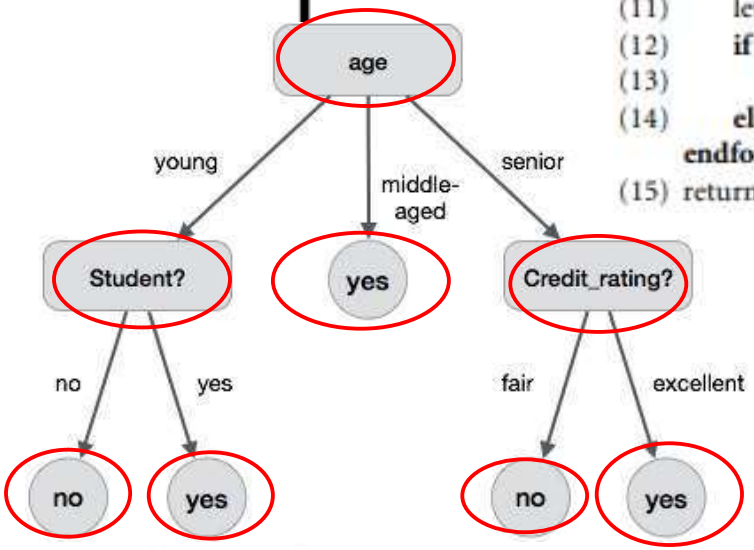


Decision Tree Basic Algorithm: construction and output

Output: A decision tree.

Method:

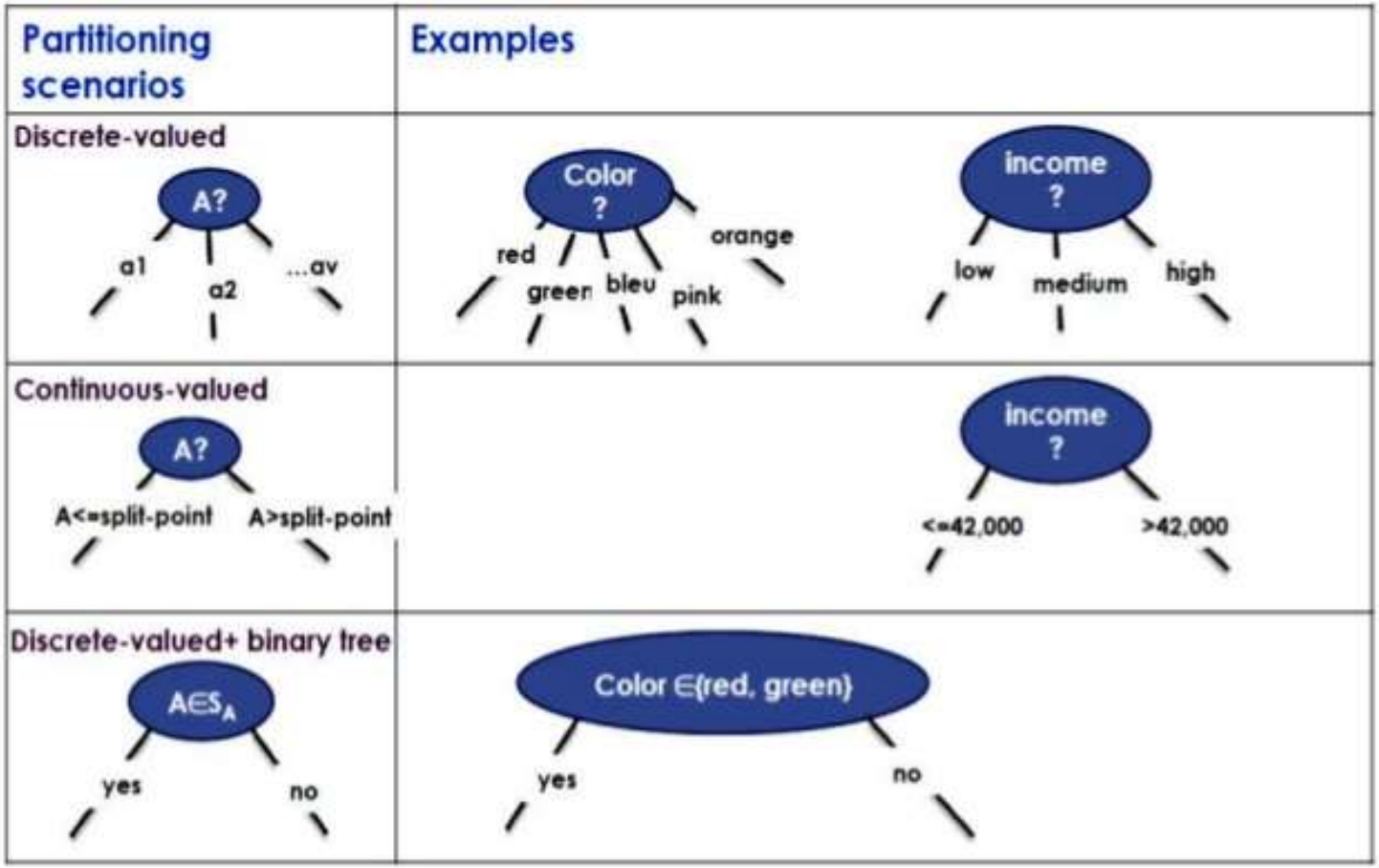
(1) create a node N;
(2) **if** tuples in D are all of the same class, C, **then**
(3) return N as a leaf node labeled with the class C;
(4) **if** *attribute_list* is empty **then**
(5) return N as a leaf node labeled with the majority class in D; // majority voting
(6) apply **Attribute_selection_method**(D, *attribute_list*) to **find** the "best" *splitting_criterion*;
(7) label node N with *splitting_criterion*;
(8) **if** *splitting_attribute* is discrete-valued **and**
multiway splits allowed **then** // not restricted to binary trees
(9) *attribute_list* ← *attribute_list* − *splitting_attribute*; // remove *splitting_attribute*
(10) **for each** outcome j of *splitting_criterion*
// partition the tuples and grow subtrees for each partition
(11) let D_j be the set of data tuples in D satisfying outcome j; // a partition
(12) **if** D_j is empty **then**
(13) attach a leaf labeled with the majority class in D to node N;
(14) **else** attach the node returned by **Generate_decision_tree**(D_j, *attribute_list*) to node N;
endfor
(15) return N;



Buy_computer decision tree

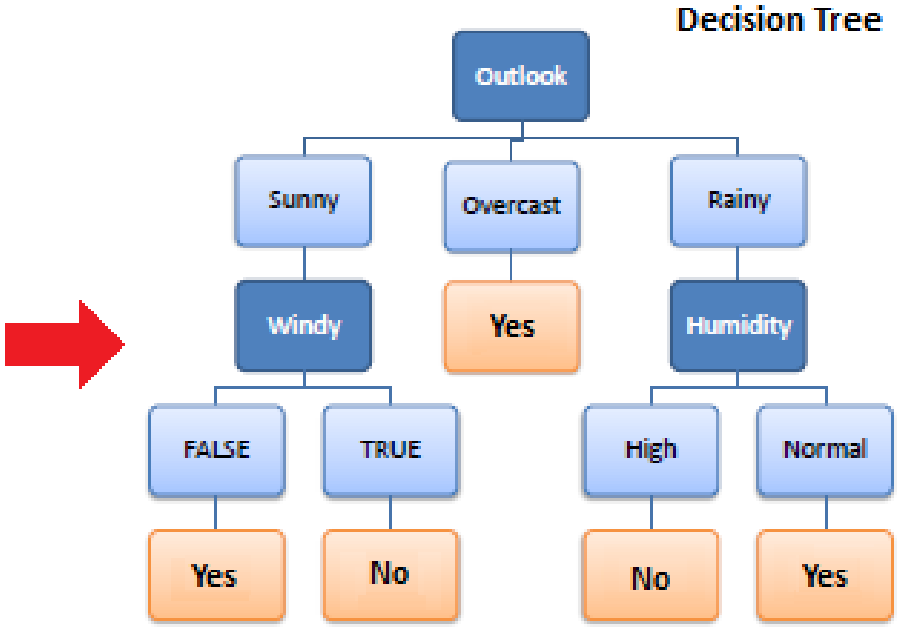
Partitioning

Three Possible Partition Scenarios



Think (Can you do another DT?)

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



Attribute Select Measures

- An attribute selection measure is a heuristic for selecting the splitting criterion that “best” separates a given data partition, D, of class labelled training tuples into individual classes.
- Ideally – Each resulting partition would be pure – A pure partition is a partition containing tuples that all belong to the same class.
- Attribute selection measures (splitting rules) – Determine how the tuples at a given node are to be split – Provide ranking for each attribute describing the tuples – The attribute with highest score is chosen – Determine a split point or a splitting subset
- There are around 30 different methods for attribute select measures (please see a benchmark evaluation report about their performance in reading materials section)

Well known measuring methods

Information Gain

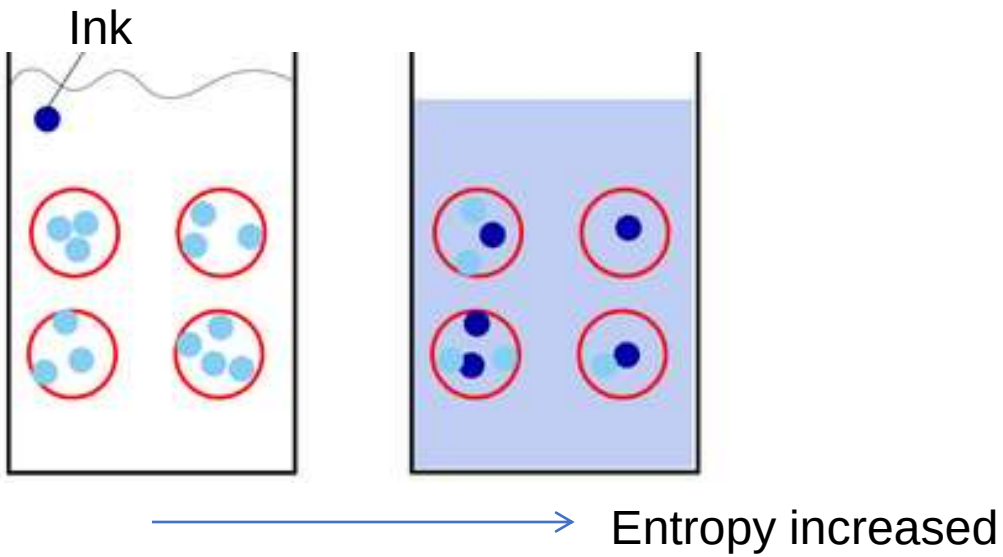
Gain Ratio

Gini Index

Before introducing the methods, we need to know a definition: “Entropy” - Shannon's metric of "Entropy" of information is a foundational concept of information theory.

Entropy in physics

In statistical mechanics, **entropy** is an extensive property of a thermodynamic system. It quantifies the number Ω of microscopic configurations (known as microstates) that are consistent with the macroscopic quantities that characterize the system (such as its volume, pressure and temperature). Wiki



Entropy in information theory

Entropy is a measure of the average information content one is missing when one does not know the value of the random variable.

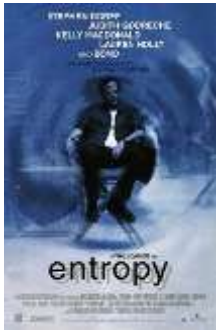
- Shannon's metric of "Entropy" of information is a foundational concept of information theory.
- The entropy of a variable is the "amount of information" contained in the variable.

Tenet film 2020

High Entropy

- X is from a uniform like distribution
- Flat histogram
- Values sampled from it are less predictable

Entropy 1999



Low Entropy

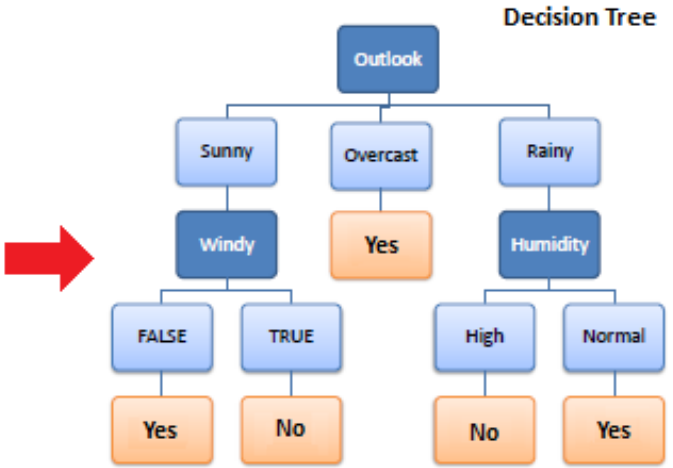
- X is from a varied (peaks and valleys) distribution
- Histogram has many lows and highs
- Values sampled from it are more predictable

Information Gain Method

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 attribute that returns the highest information gain (i.e., the most homogeneous branches).

It has 2 steps

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



First step: 2-types of entropy computation

a) Entropy using the frequency table of one attribute:

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

A math explanation

Play Golf	
Yes	No
9	5

5:14

Entropy(PlayGolf) = Entropy (5,9)

= Entropy (0.36, 0.64)

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

= 0.94

2-types of entropy computation

b) Entropy using the frequency table of two attributes:

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

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

E(PlayGolf, Outlook) = **P**(Sunny)*__E__(3,2) + **P**(Overcast)*__E__(4,0) + **P**(Rainy)*__E__(2,3)

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

= 0.693

Second Step: Information Gain Computation steps:

Step 1: Calculate entropy of the target:

Entropy(PlayGolf) = Entropy (5,9)

= Entropy (0.36, 0.64)

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

= 0.94

Step 2 : 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.

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

G(PlayGolf, Outlook) = **E**(PlayGolf) – **E**(PlayGolf, Outlook)

= 0.940 – 0.693 = 0.247

Information Gain Steps:

Step 3: 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.

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

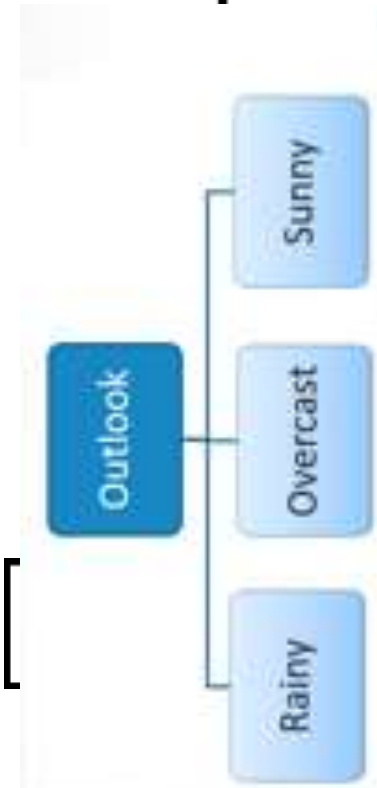
		Play Golf	
		Yes	No
Temp.	Hot	2	2
	Mild	4	2
	Cool	3	1
Gain = 0.029			

		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			

Information Gain Steps:

Step 4: loop the process to find all leaf nodes

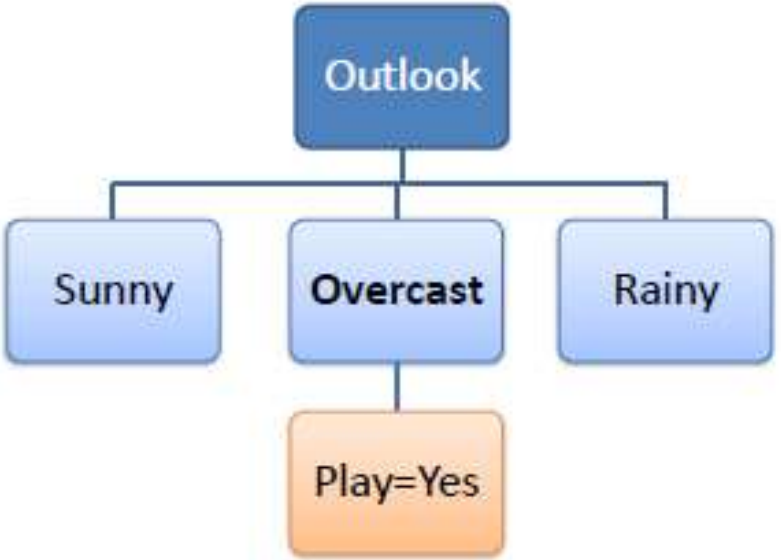


Outlook	Temp	Humidity	Windy	Play Golf
Sunny	Mild	High	FALSE	Yes
Sunny	Cool	Normal	FALSE	Yes
Sunny	Cool	Normal	TRUE	No
Sunny	Mild	Normal	FALSE	Yes
Sunny	Mild	High	TRUE	No
Overcast	Hot	High	FALSE	Yes
Overcast	Cool	Normal	TRUE	Yes
Overcast	Mild	High	TRUE	Yes
Overcast	Hot	Normal	FALSE	Yes
Rainy	Hot	High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

Information Gain Steps:

Step 4a: A branch with entropy of 0 is a leaf node.

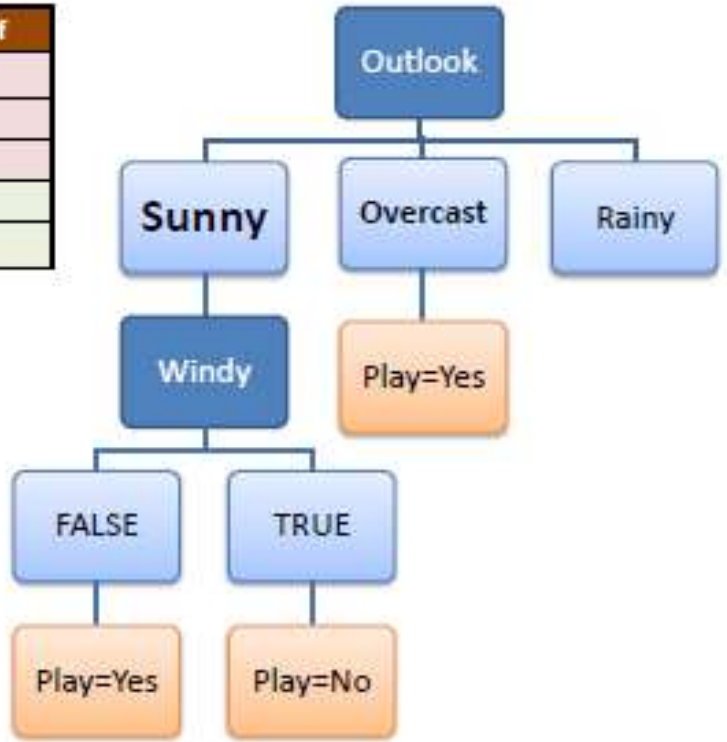
Temp	Humidity	Windy	Play Golf
Hot	High	FALSE	Yes
Cool	Normal	TRUE	Yes
Mild	High	TRUE	Yes
Hot	Normal	FALSE	Yes



Information Gain Steps:

Step 4b: A branch with entropy more than 0 needs further splitting.

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



Information Gain steps:

Finalised the decision tree and it can be easily transformed to become rules

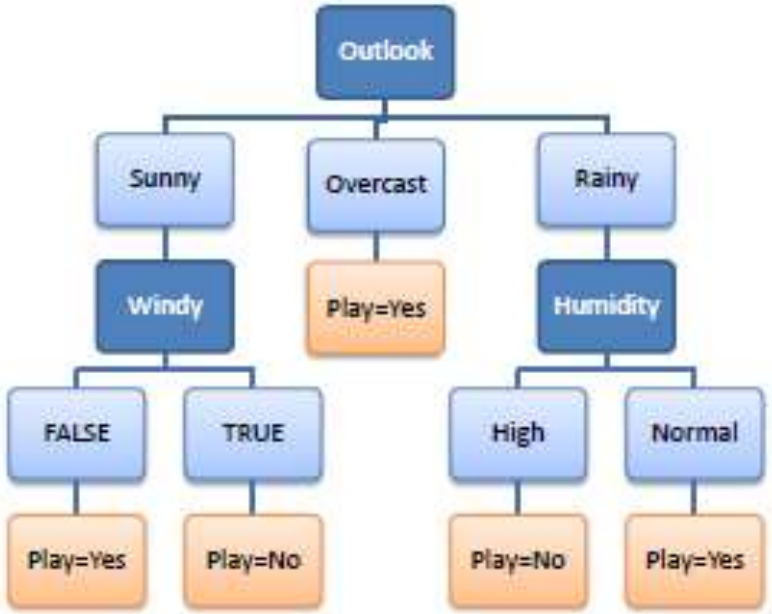
R_1: IF (Outlook=Sunny) AND (Windy=FALSE) THEN Play=Yes

R_2: IF (Outlook=Sunny) AND (Windy=TRUE) THEN Play=No

R_3: IF (Outlook=Overcast) THEN Play=Yes

R_4: IF (Outlook=Rainy) AND (Humidity=High) THEN Play=No

R_5: IF (Outlook=Rain) AND (Humidity=Normal) THEN Play=Yes



ID3

ID3 is an implementation decision tree constructive algorithm based completely on information gain process

1. Use existed ID3 library:

pip install decision-tree-id3

```
>>> from sklearn.datasets import load_breast_cancer
>>> from id3 import Id3Estimator
>>> from id3 import export_graphviz
>>> bunch = load_breast_cancer()
>>> estimator = Id3Estimator()
>>> estimator.fit(bunch.data, bunch.target)
>>> export_graphviz(estimator.tree_, 'tree.dot', bunch.feature_names)
```

2. Implement ID3 yourself:

https://medium.com/@lope.ai/decision-trees-from-scratch-using-id3-python-coding-it-up-6b79e3458de4

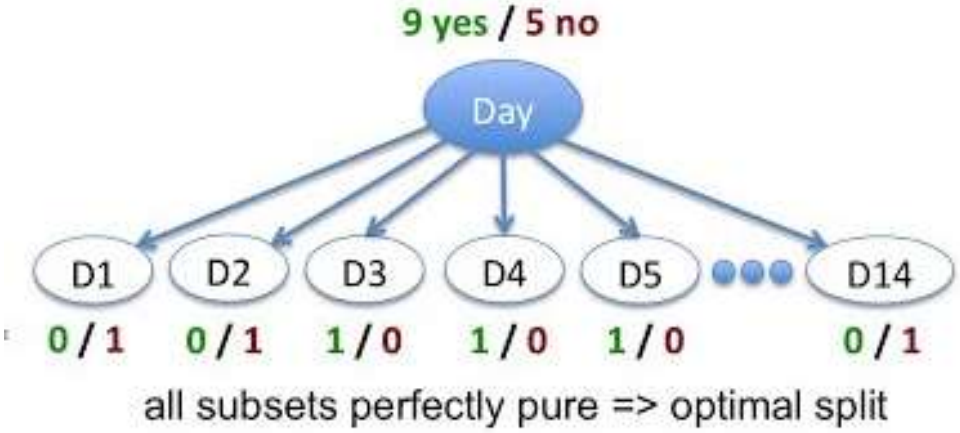
Issue of information gain

Information gain is a good measure but it has some flaws that we need to understand

If we split our dataset on Day attribute then we ended with the singleton subsets. That is on each day will either play or not.

E.g. D=15?

Not work



Gain Ratio

Punishing the many splitting attributes with gain ration

$$SplitEntropy(S,A) = -\sum_{V \in Values(A)} \frac{|S_V|}{|S|} \log \frac{|S_V|}{|S|}$$

A … candidate attribute
V … possible values of A
S … set of examples {X}
S_v … subset where $X_A = V$

$$GainRatio(S,A) = \frac{Gain(S,A)}{SplitEntropy(S,A)}$$

penalizes attributes with many values

Gain Ratio (Cont.)

$$SplitEntropy(S,A) = -\sum_{V \in Values(A)} \frac{|S_V|}{|S|} \log \frac{|S_V|}{|S|}$$

A … candidate attribute
V … possible values of A
S … set of examples {X}
S_v … subset where $X_A = V$

$$GainRatio(S,A) = \frac{Gain(S,A)}{SplitEntropy(S,A)}$$

penalizes attributes with many values

Entropy (S, A)

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

E(PlayGolf, Outlook) = **P**(Sunny)***E**(3,2) + **P**(Overcast)***E**(4,0) + **P**(Rainy)***E**(2,3)

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

= 0.693

Gain (S, A) = Entropy (S)- Entropy(S,A)

G(PlayGolf, Outlook) = **E**(PlayGolf) – **E**(PlayGolf, Outlook)

= 0.940 – 0.693 = 0.247

SplitEntropy (S, A)
= -5/14*log(5/14)+ -4/14*log(4/14)+ -5/14*log(5/14)
= 1.577

GainRatio (S, A)
= Gain(S, A)/SplitEntropy(S, A)
=0.247/1.577
= 0.157

Gain Ratio results

Outlook		Temperature	
Info:	0.693	Info:	0.911
Gain: 0.940-0.693	0.247	Gain: 0.940-0.911	0.029
Split info: info([5,4,5])	1.577	Split info: info([4,6,4])	1.557
Gain ratio: 0.247/1.577	0.157	Gain ratio: 0.029/1.557	0.019
Humidity		**Windy**	
Info:	0.788	Info:	0.892
Gain: 0.940-0.788	0.152	Gain: 0.940-0.892	0.048
Split info: info([7,7])	1.000	Split info: info([8,6])	0.985
Gain ratio: 0.152/1	0.152	Gain ratio: 0.048/0.985	0.049

Gain Ratio potential issue and C4.5

It may overcompensate. May choose an attribute just because its intrinsic information is very low.

Standard fix: only consider attributes with greater than average information gain

C4.5 is the implemented Gain Ration algorithm, a successor of ID3 with more functions of deals with numeric attributes, missing values and noisy data.

Gini Index

In computation, entropy is not preferred due to the 'log' function as it increases the computational complexity.

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

Gini index try to use a more simpler computation to do the measurement. But Gini index considers a binary split for each attribute.

$$\text{Gini}_A(D) = \frac{|D1|}{|D|} Gini(D1) + \frac{|D2|}{|D|} Gini(D2)$$

Final impurity = Gini (D) - $\text{Gini}_A(D)$

Gini Index example

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

1. Play = 9 days, not play = 5 days

Gini (D) = 1- $(9/14)^2$ – $(5/14)^2$ = 0.459

2.
Rainy (5 days):
Play = 2 days, not play = 3 days
Not_rainy:
play (9 days) = 7 days, not play = 2 days

$$Gini_A(D) = \frac{|D1|}{|D|} Gini(D1) + \frac{|D2|}{|D|} Gini(D2)$$

$$Gini_{rainy}(D) = 5/14(Gini(D1))+9/14(Gini(D2))$$
$$=5/14(1-(2/5)^2- (3/5)^2) + 9/14(1-(7/9)^2- (2/9)^2)$$
$$= 0.1786 + 0.2218$$
$$= 0.4004$$

3. Gini (D) - $Gini_A(D)$
= 0.459 – 0.4004 = 0.055
= impurity

$$Gini = 1 - \sum_{i=1}^{n} (p_i)^2$$

Implementation of Gini Index is CART algorithm

Other attribute selection measures

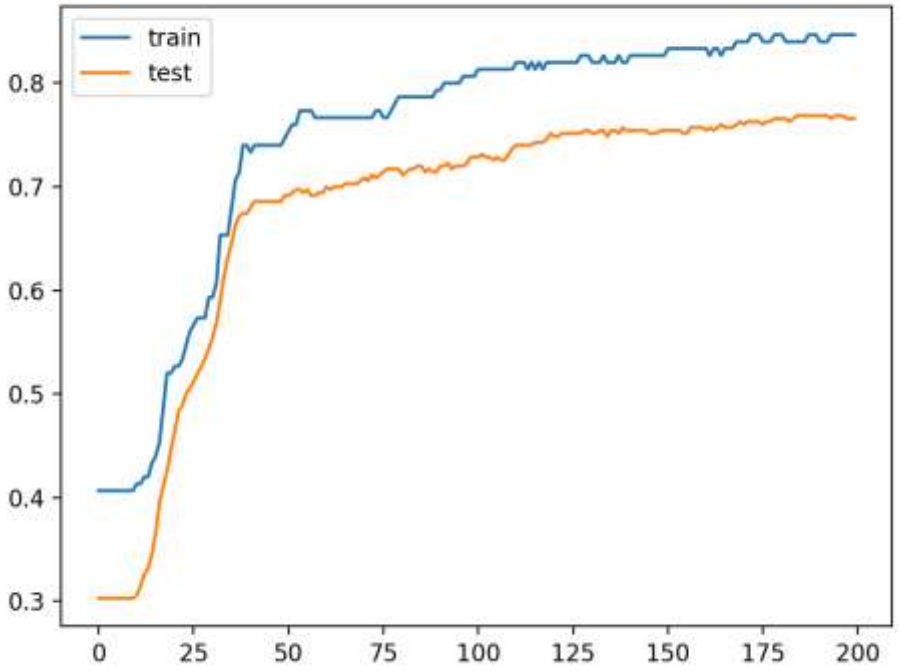
Decision Tree Regression approach

- In regression the task instead of classifying point to one class we calculate the real value.
- Unlike calculating the information gain we calculate the variance and try to minimize it. So we split data into multiple subsets based on attributes and took one with minimum variance.
- While testing we reach the particular leaf node and took the average over all the points in that subsets and assign the average value to that testing point.

MDL (Minimum Description Length) approach

Avoiding overfitting in decision tree

Overfitting is a significant practical difficulty for decision tree models and many other predictive models. Overfitting happens when the learning algorithm continues to develop hypotheses that reduce training set error at the cost of an increased test set error.



Tree Pruning

Prevent overfitting to noise in the data

Two strategies:

- Post-pruning: take a fully-grown decision tree and discard unreliable parts
- Pre-pruning: stop growing a branch when information becomes unreliable

In practice, the post-pruning is the more common used strategy

UNIVERSITY OF DERBY

Self-study part for your interest

TOP 30 GUARDIAN UNIVERSITY GUIDE 2019

derby.ac.uk

Post-pruning in detail

In post-pruning, we take training data and divide it into 2 parts. Hide the one part of data completely from the algorithm that we called as cross-validation (CV) and train a full decision tree until we get leaf nodes.

Now take out validation set and test on the learned tree which will not give 100% accuracy because the model has overfitted.

After this for each non-leaf node in the tree, we temporarily prune the node and replace it by majority class. Again test the validation set on the pruned tree we get another accuracy.

Repeat the same procedure for each non-leaf node and record the validation accuracy and took one pruned tree which will give the highest accuracy on validation data.

Partition the train data into 2 parts.

1. Build the complete tree on train data.
2. Until accuracy on cross-validation increases do

{

for each non-leaf node

1. Temporarily prune it and replace by majority vote.
2. Test the accuracy of the model on CV data.
3. Permanently prune the node with the greatest accuracy on the CV.

}

Error estimate function:

$$e = \left(f + \frac{z^2}{2N} + z\sqrt{\frac{f}{N} - \frac{f^2}{N} + \frac{z^2}{4N^2}} \right) \Big/ \left(1 + \frac{z^2}{N} \right)$$

Where:

- f is the error on the training data
- N is the number of instances covered by the leaf
- z from normal distribution

The other error estimate method called Chi^2 test
https://www.saedsayad.com/categorical_categorical.htm

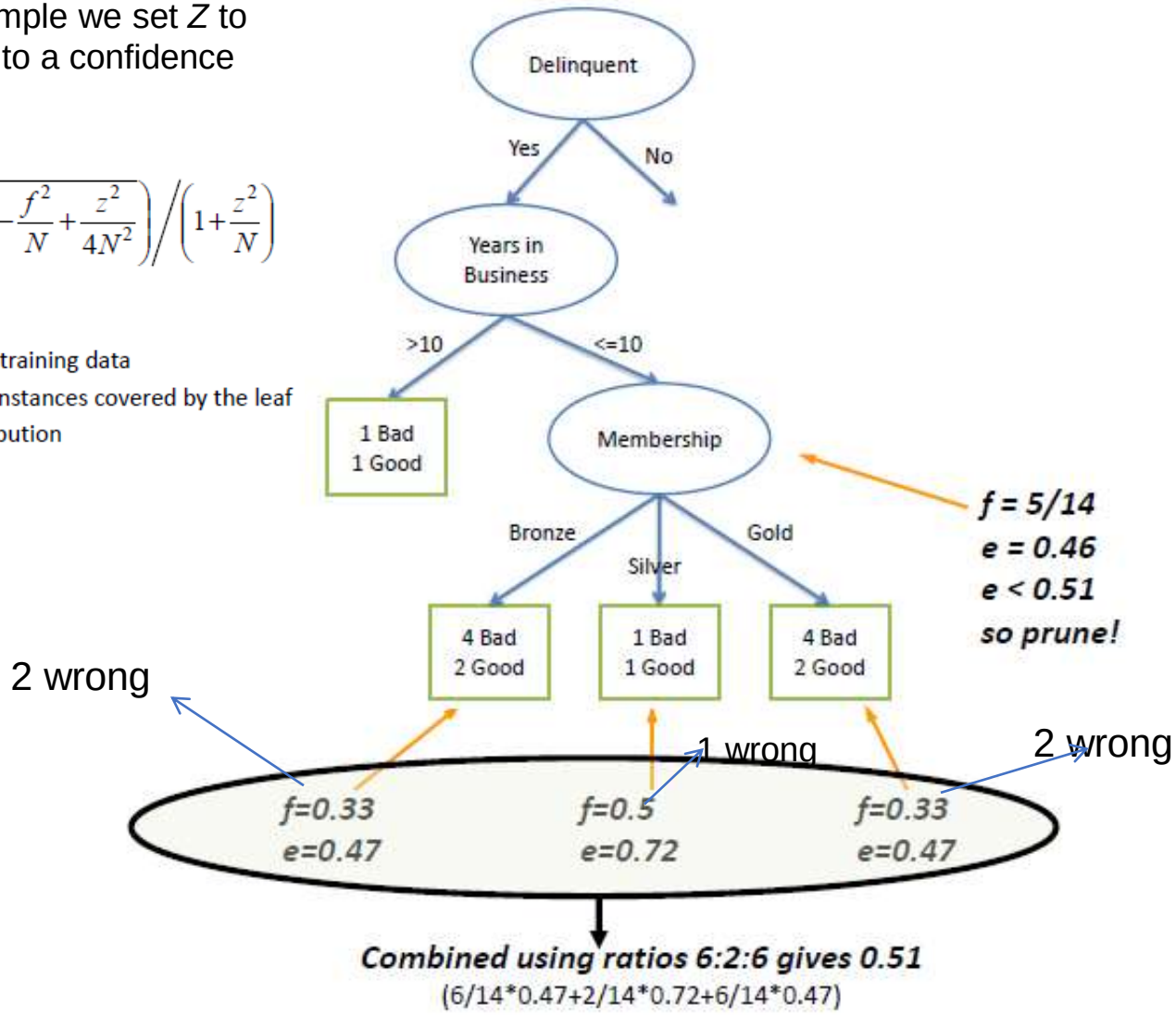
Pruning example

In the following example we set *Z* to 0.69 which is equal to a confidence level of 75%

$$e = \left(f + \frac{z^2}{2N} + z\sqrt{\frac{f}{N} - \frac{f^2}{N} + \frac{z^2}{4N^2}} \right) \Big/ \left(1 + \frac{z^2}{N} \right)$$

Where:

- f is the error on the training data
- N is the number of instances covered by the leaf
- z from normal distribution



Time and Space complexity

Training phase :

T (n) = O (nlogn * m) is a kind of NP hard problem (here is the prove explanation)

Where m = number of features

Testing phase:

S (n) = script of nested if-else condition

∴ S (n) = O (Total nodes)

T (n) = O (Depth)

Summarisation:
Pros and Cons of the decision tree

Pros:

1. DT is a sequence of if-else which is highly interpretable to humans.
2. It handles the irrelevant attributes easily.
3. DT automatically does the multiclass classification.
4. Once the DT is built it becomes very fast at the testing time.
5. DT's are very compact.

Cons:

1. The decision tree may not find always the best tree because it uses the greedy approach to build the tree.
2. For every large dimensional data, the training time is high.
3. DT only forms axis parallel hyperplanes for a real-valued attribute.

UNIVERSITY OF DERBY

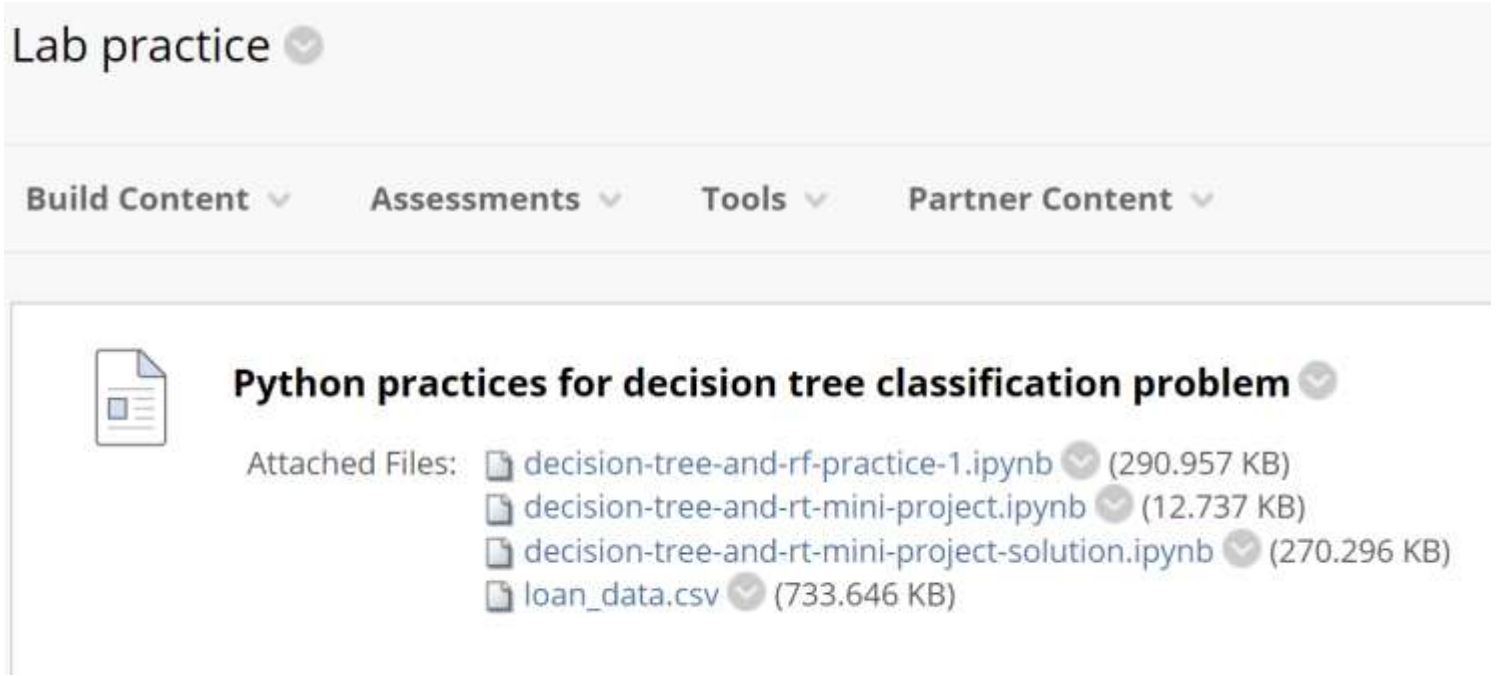
Next Week

1. Random Forest
2. K-NN

TOP 30 GUARDIAN UNIVERSITY GUIDE 2019

Now let's do DT using Python

Week 4 folder – Lab practice



UNIVERSITY OF DERBY

Online references

https://www.tutorialspoint.com/data_mining/dm_classification_prediction.htm

https://machinelearningmastery.com/types-of-classification-in-machine-learning/

https://www.softwaretestinghelp.com/decision-tree-algorithm-examples-data-mining/

https://www.aitimejournal.com/@akshay.chavan/a-comprehensive-guide-to-decision-tree-learning

https://www.saedsayad.com/decision_tree_overfitting.htm

https://machinelearningmastery.com/implement-decision-tree-algorithm-scratch-python/

https://stackabuse.com/decision-trees-in-python-with-scikit-learn/

TOP 30 GUARDIAN UNIVERSITY GUIDE 2019