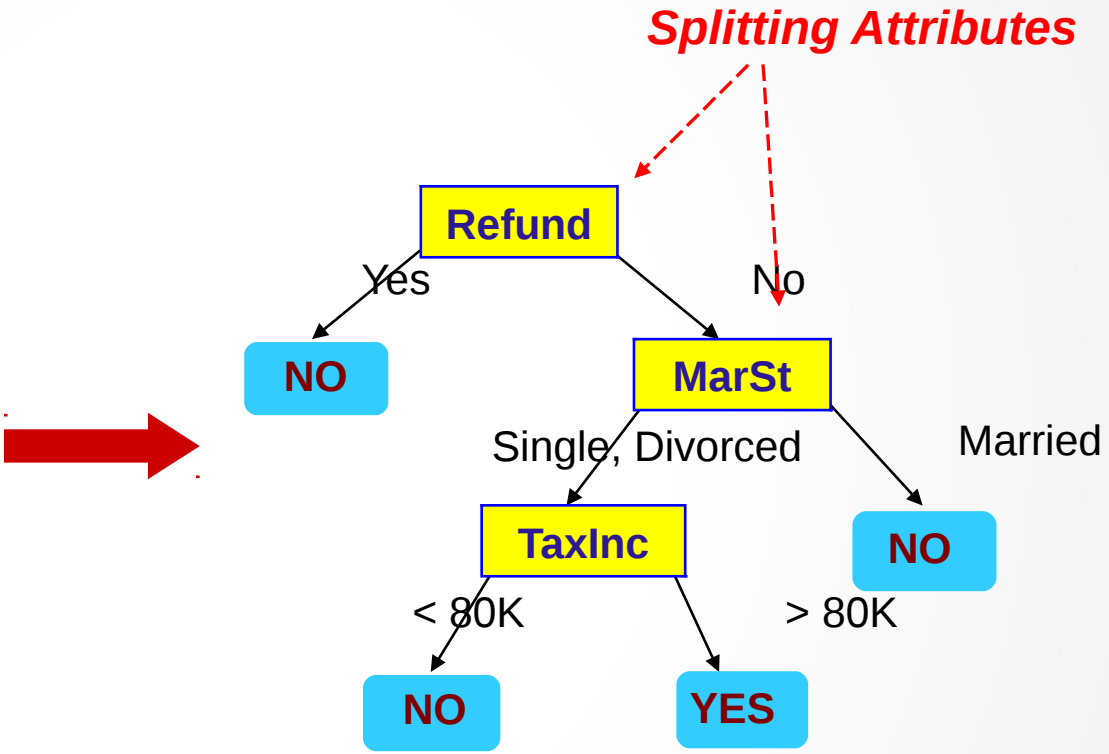
Decision Trees and Randomforest

Jony Sugianto
jony@evolvemachinelearners.com
0812-13086659
github.com/jonysugianto

Example of a Decision Tree

Tid	Refund (categorical)	Marital Status (categorical)	Taxable Income (continuous)	Cheat (class)
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

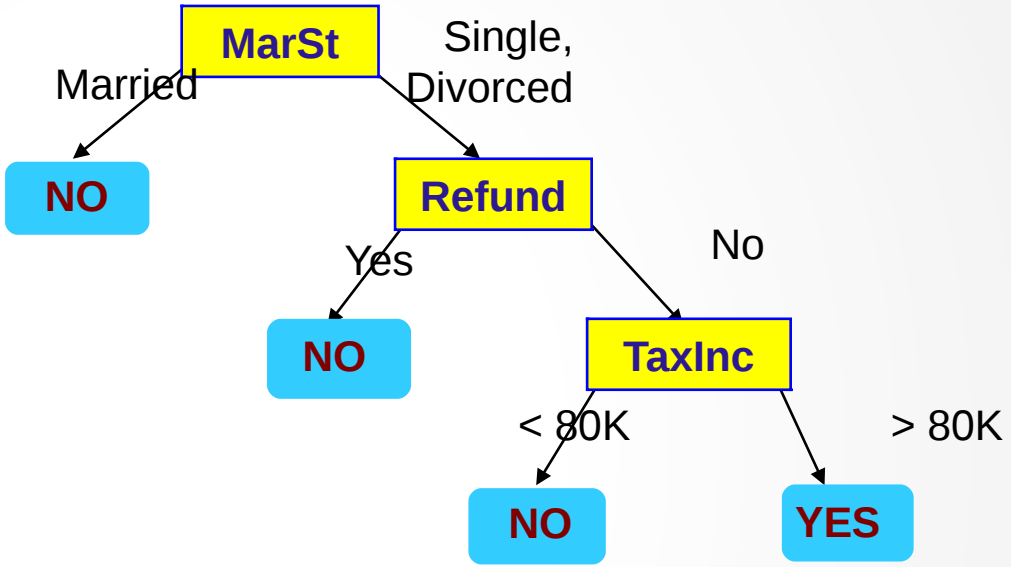
Training Data



Model: Decision Tree

Another Example of Decision Tree

Tid	Refund (categorical)	Marital Status (categorical)	Taxable Income (continuous)	Cheat (class)
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

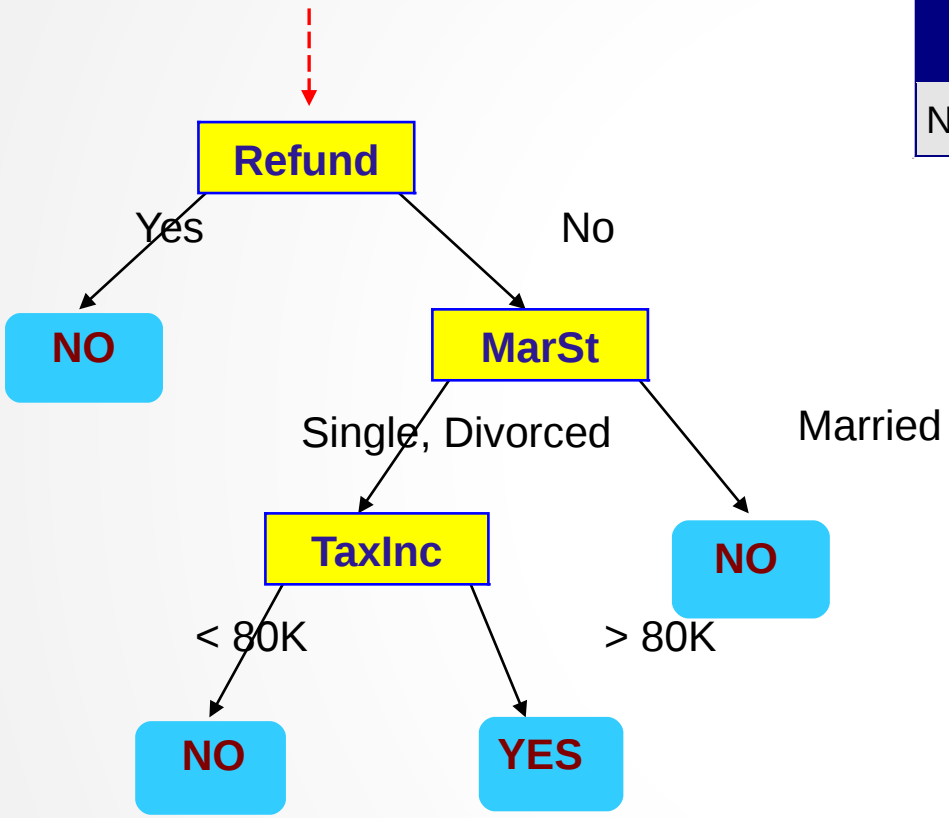


There could be more than one tree that fits the same data!

Apply Model to Test Data

EVOLVE
MACHINE LEARNERS

Start from the root of tree.



Test Data

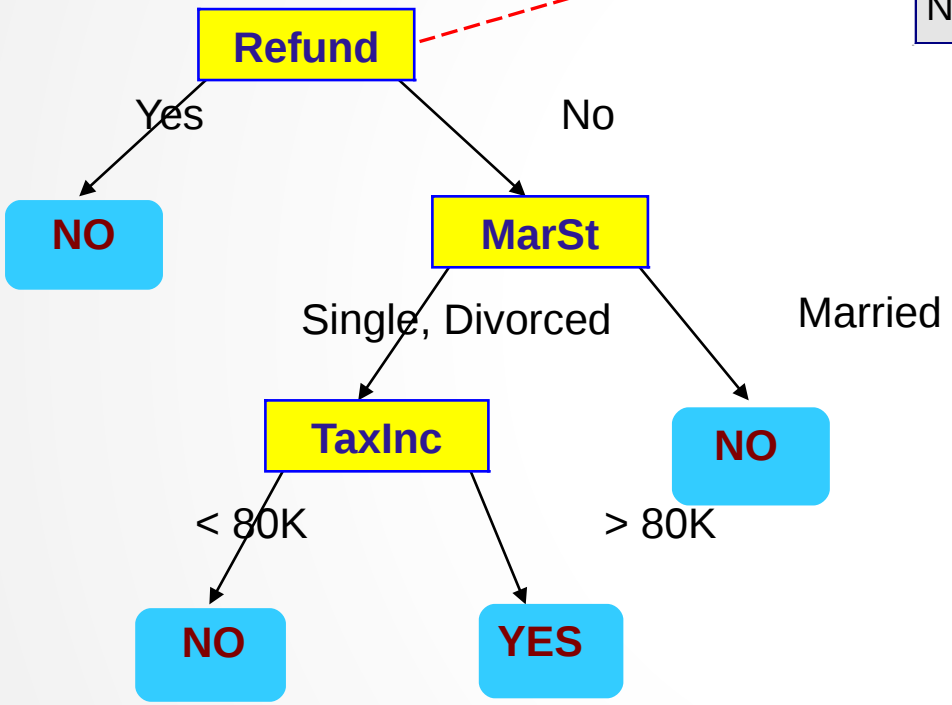
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data

EVOLVE
MACHINE LEARNERS

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

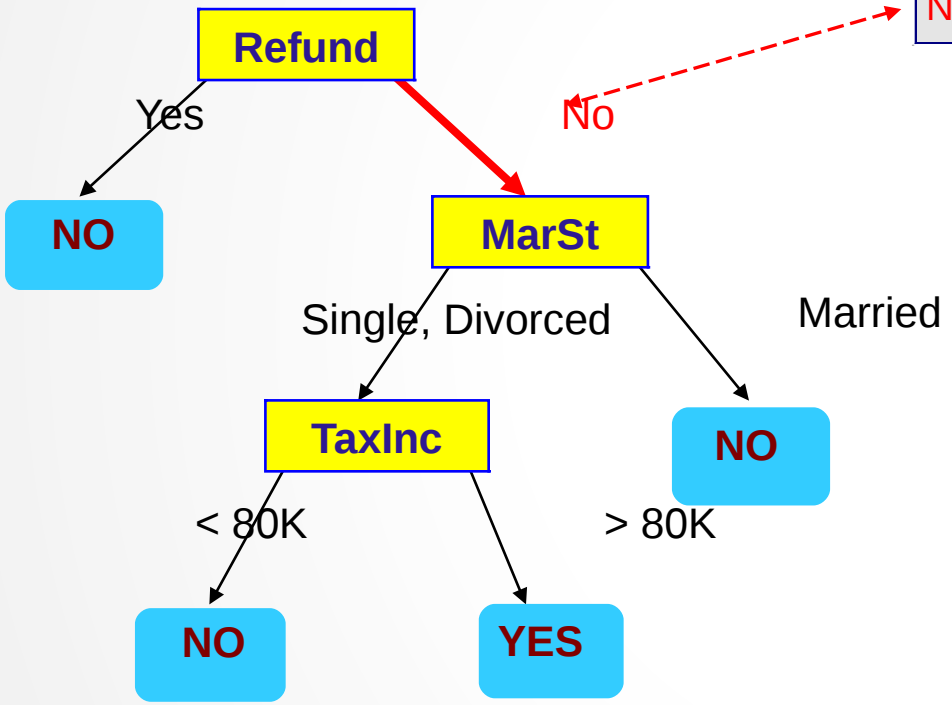


Apply Model to Test Data

EVOLVE
MACHINE LEARNERS

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

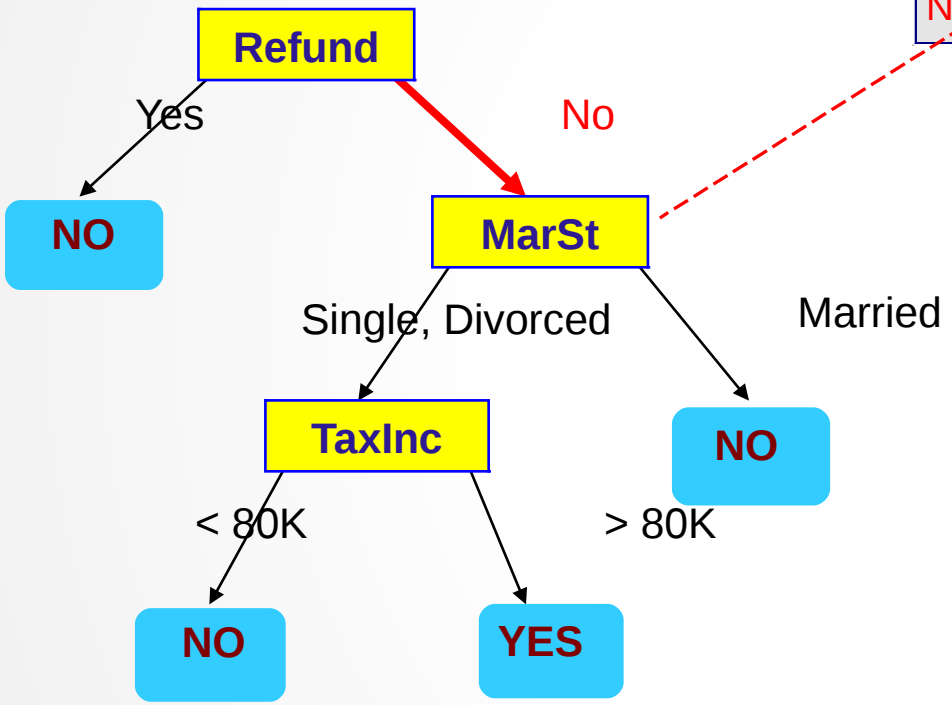


Apply Model to Test Data

EVOLVE
MACHINE LEARNERS

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

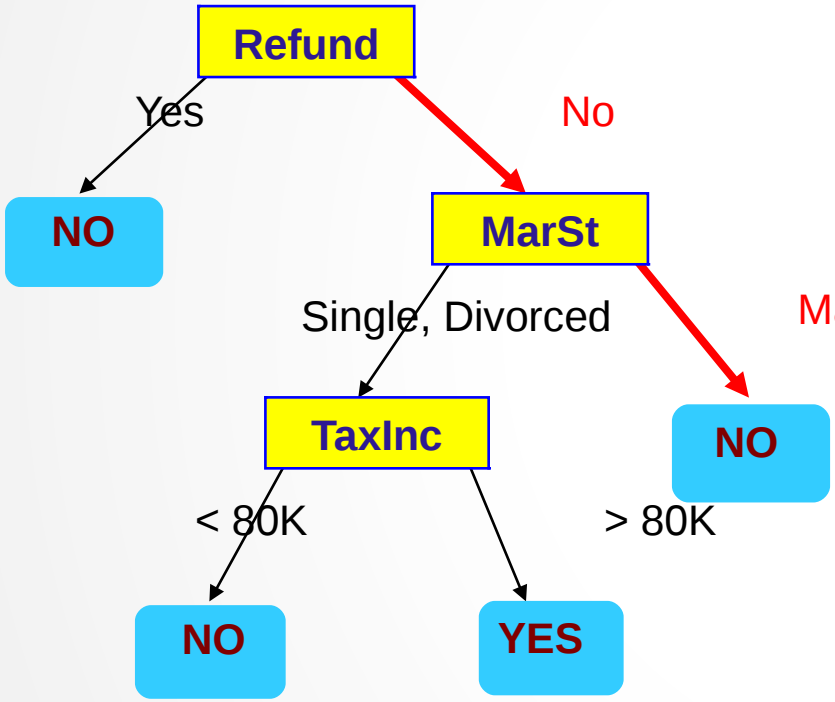


Apply Model to Test Data

EVOLVE
MACHINE LEARNERS

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

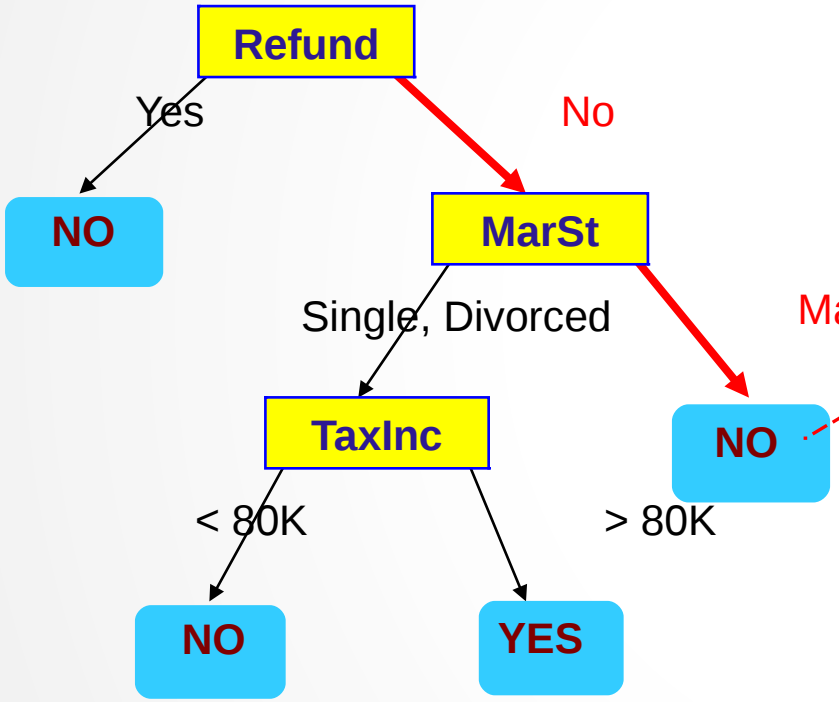


Apply Model to Test Data

EVOLVE
MACHINE LEARNERS

Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Assign Cheat to “No”

Tree Induction

- Greedy strategy.
  - Split the records based on an attribute test that optimizes certain criterion.

- Issues
  - Determine how to split the records
    - How to specify the attribute test condition?
    - How to determine the best split?
  - Determine when to stop splitting

Tree Induction

- Greedy strategy.
  - Split the records based on an attribute test that optimizes certain criterion.

- Issues
  - Determine how to split the records
    - How to specify the attribute test condition?
    - How to determine the best split?
  - Determine when to stop splitting

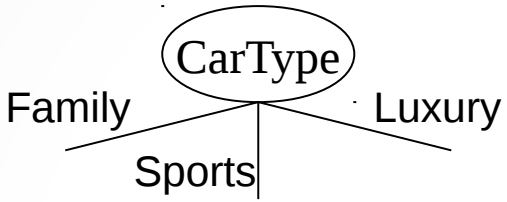
How to Specify Test Condition?

- Depends on attribute types
  - Nominal
  - Ordinal
  - Continuous

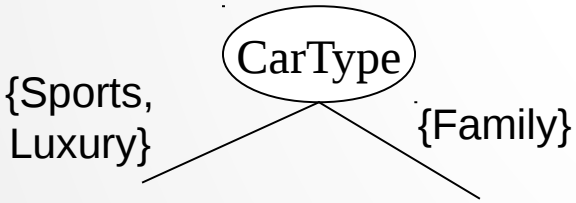
- Depends on number of ways to split
  - 2-way split
  - Multi-way split

Splitting Based on Nominal Attributes

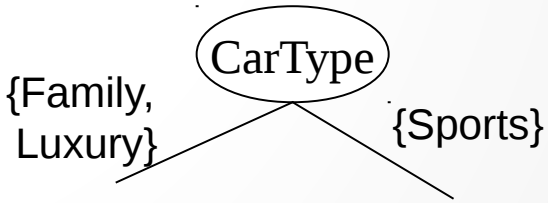
- Multi-way split: Use as many partitions as distinct values.



- Binary split: Divides values into two subsets. Need to find optimal partitioning.

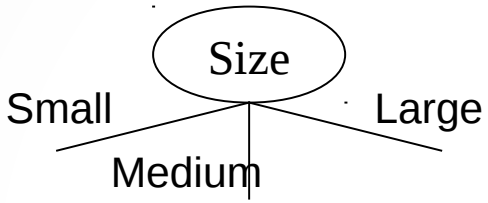


OR

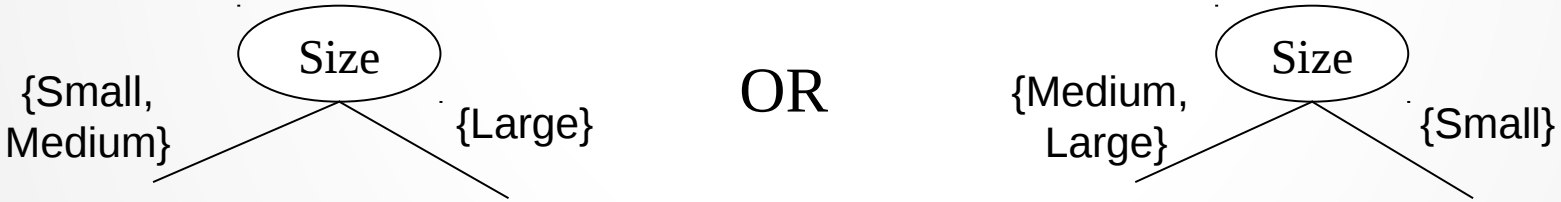


Splitting Based on Ordinal Attributes

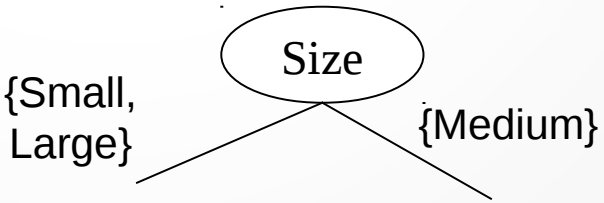
- Multi-way split: Use as many partitions as distinct values.



- Binary split: Divides values into two subsets. Need to find optimal partitioning.



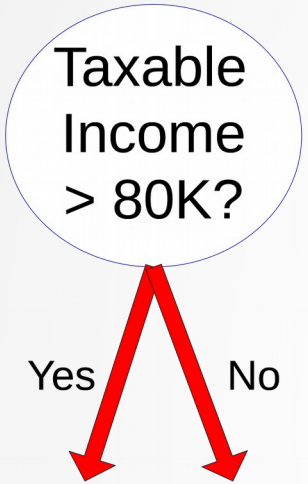
- What about this split?



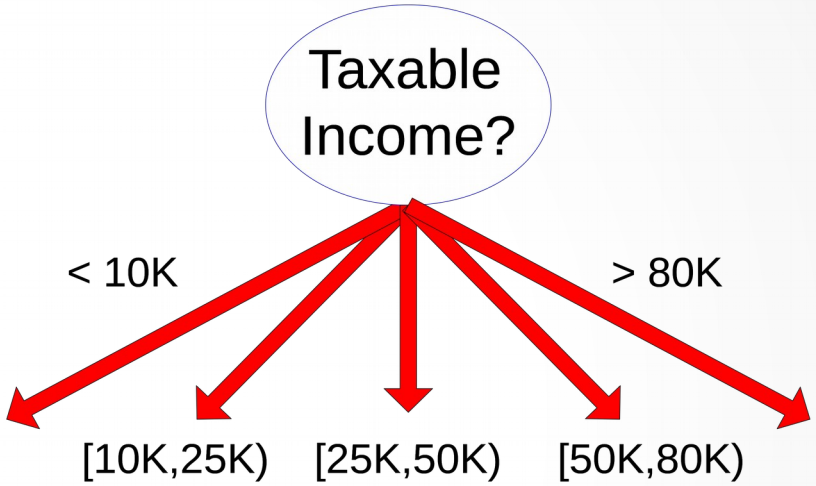
Splitting Based on Continuous Attributes

- Different ways of handling
  - Discretization to form an ordinal categorical attribute
    - Static – discretize once at the beginning
    - Dynamic – ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles), or clustering.
  - Binary Decision: $(A < v)$ or $(A \geq v)$
    - consider all possible splits and finds the best cut
    - can be more compute intensive

Splitting Based on Continuous Attributes



(i) Binary split



(ii) Multi-way split

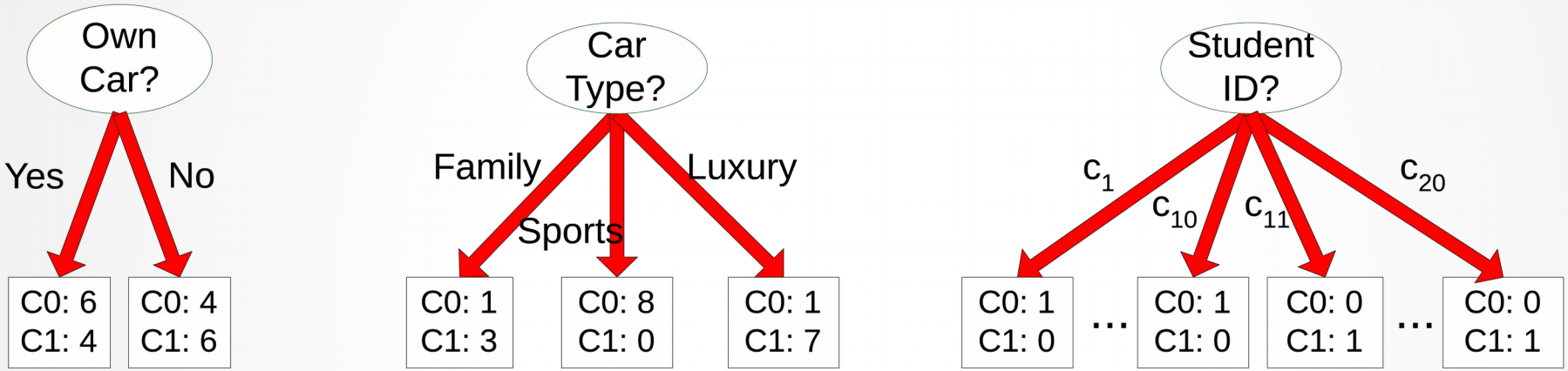
Tree Induction

- Greedy strategy.
  - Split the records based on an attribute test that optimizes certain criterion.

- Issues
  - Determine how to split the records
    - How to specify the attribute test condition?
    - How to determine the best split?
  - Determine when to stop splitting

How to determine the Best Split

Before Splitting: 10 records of class 0,
10 records of class 1



Which test condition is the best?

How to determine the Best Split

EVOLVE
MACHINE LEARNERS

- Greedy approach:
  - Nodes with homogeneous class distribution are preferred
- Need a measure of node impurity:

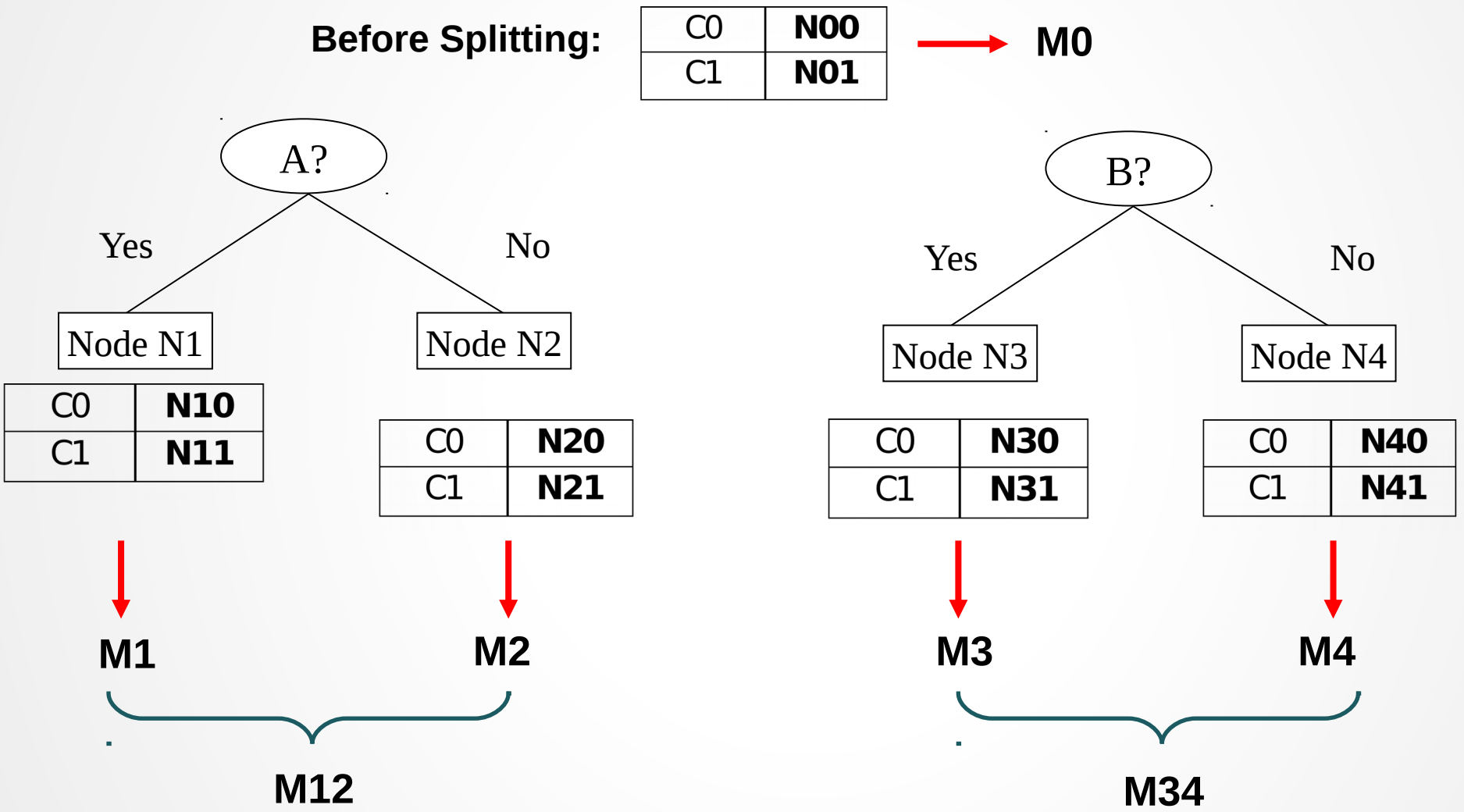
C0: 5 C1: 5	C0: 9 C1: 1
Non-homogeneous, **High degree of impurity**	**Homogeneous,** **Low degree of impurity**

Measures of Node Impurity

EVOLVE
MACHINE LEARNERS

- Gini Index
- Entropy
- Misclassification error

How to Find the Best Split



Measure of Impurity: GINI

EVOLVE
MACHINE LEARNERS

- Gini Index for a given node t :

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

(NOTE: $p(j \mid t)$ is the relative frequency of class j at node t).

  - Maximum (1 - $1/n_c$) when records are equally distributed among all classes, implying least interesting information
  - Minimum (0.0) when all records belong to one class, implying most interesting information

C1	**0**
C2	**6**
Gini=0.000	

C1	**1**
C2	**5**
Gini=0.278	

C1	**2**
C2	**4**
Gini=0.444	

C1	**3**
C2	**3**
Gini=0.500	

Examples for computing GINI

$$GINI(t) = 1 - \sum_j [p(j \mid t)]^2$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Gini = 1 – P(C1)² – P(C2)² = 1 – 0 – 1 = 0

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

Gini = 1 – (1/6)² – (5/6)² = 0.278

C1	**2**
C2	**4**

P(C1) = 2/6 P(C2) = 4/6

Gini = 1 – (2/6)² – (4/6)² = 0.444

Splitting Based on GINI

- Used in CART, SLIQ, SPRINT.
- When a node p is split into k partitions (children), the quality of split is computed as,

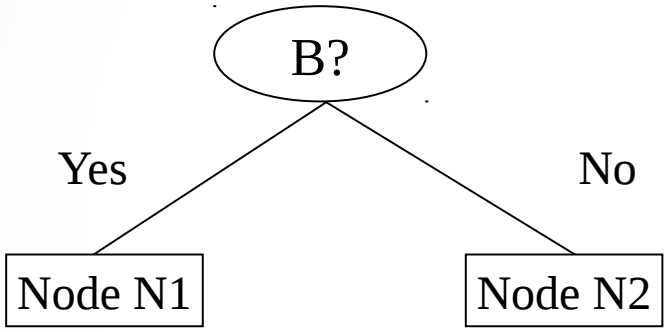
$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

where, n_i = number of records at child i,

n = number of records at node p.

Binary Attributes: Computing GINI Index

- Splits into two partitions
- Effect of Weighing partitions:
  - Larger and Purer Partitions are sought for.



	Parent
C1	**6**
C2	**6**
Gini = 0.500	

Gini(N1)
= 1 – (5/6)² – (2/6)²
= 0.194

Gini(N2)
= 1 – (1/6)² – (4/6)²
= 0.528

	N1	N2
C1	**5**	**1**
C2	**2**	**4**
Gini=0.333		

Gini(Children)
= 7/12 * 0.194 +
5/12 * 0.528
= 0.333

Categorical Attributes: Computing Gini Index

EVOLVE
MACHINE LEARNERS

- For each distinct value, gather counts for each class in the dataset
- Use the count matrix to make decisions

Multi-way split

	CarType		
	Family	Sports	Luxury
C1	1	2	1
C2	4	1	1
Gini	0.393		

Two-way split
(find best partition of values)

	CarType	
	{Sports, Luxury}	{Family}
C1	3	1
C2	2	4
Gini	0.400	

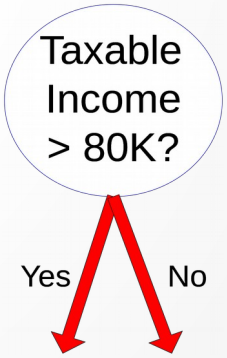
	CarType	
	{Sports}	{Family, Luxury}
C1	2	2
C2	1	5
Gini	0.419	

Continuous Attributes: Computing Gini Index

- Use Binary Decisions based on one value
- Several Choices for the splitting value
  - Number of possible splitting values

    = Number of distinct values
- Each splitting value has a count matrix associated with it
  - Class counts in each of the partitions, A < v and A ≥ v
- Simple method to choose best v
  - For each v, scan the database to gather count matrix and compute its Gini index
  - Computationally Inefficient! Repetition of work.

Tid	Refund	Marital Status	Taxable Income	Cheat
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



Continuous Attributes: Computing Gini Index...

EVOLVE
MACHINE LEARNERS

- For efficient computation: for each attribute,
  - Sort the attribute on values
  - Linearly scan these values, each time updating the count matrix and computing gini index
  - Choose the split position that has the least gini index

Cheat	No	No	No	Yes	Yes	Yes	No	No	No	No
Taxable Income										
Sorted Values →	60	70	75	85	90	95	100	120	125	220
Split Positions →	55	65	72	80	87	92	97	110	122	172	230

	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>	<=	>
Yes	0	3	0	3	0	3	0	3	1	2	2	1	3	0	3	0	3	0	3	0	3	0
No	0	7	1	6	2	5	3	4	3	4	3	4	3	4	4	3	5	2	6	1	7	0
Gini	0.420		0.400		0.375		0.343		0.417		0.400		***0.300***		0.343		0.375		0.400		0.420	

Alternative Splitting Criteria based on INFO

- Entropy at a given node t:

$$Entropy(t) = -\sum_j p(j \mid t) \log p(j \mid t)$$

(NOTE: $p(j \mid t)$ is the relative frequency of class j at node t).

  - Measures homogeneity of a node.
    - Maximum ($\log n_c$) when records are equally distributed among all classes implying least information
    - Minimum (0.0) when all records belong to one class, implying most information
  - Entropy based computations are similar to the GINI index computations

Examples for computing Entropy

$$Entropy(t) = -\sum_j p(j|t)\log_2 p(j|t)$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Entropy = – 0 log 0 – 1 log 1 = – 0 – 0 = 0

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

Entropy = – (1/6) $\log_2$ (1/6) – (5/6) $\log_2$ (1/6) = 0.65

C1	**2**
C2	**4**

P(C1) = 2/6 P(C2) = 4/6

Entropy = – (2/6) $\log_2$ (2/6) – (4/6) $\log_2$ (4/6) = 0.92

Splitting Based on INFO...

- Information Gain:

$$GAIN_{split} = Entropy(p) - \left[\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i) \right]$$

Parent Node, p is split into k partitions;

n_i is number of records in partition i

  - Measures Reduction in Entropy achieved because of the split. Choose the split that achieves most reduction (maximizes GAIN)
  - Used in ID3 and C4.5
  - Disadvantage: Tends to prefer splits that result in large number of partitions, each being small but pure.

Splitting Based on INFO...

- Gain Ratio:

$$GainRATIO_{split} = \frac{GAIN_{Split}}{SplitINFO} \qquad SplitINFO = -\sum_{i=1}^{k} \frac{n_i}{n} \log \frac{n_i}{n}$$

Parent Node, p is split into k partitions

n_i is the number of records in partition i

- Adjusts Information Gain by the entropy of the partitioning (SplitINFO). Higher entropy partitioning (large number of small partitions) is penalized!
- Used in C4.5
- Designed to overcome the disadvantage of Information Gain

Splitting Criteria based on Classification Error

- Classification error at a node t :

$$Error(t) = 1 - \max_i P(i \mid t)$$

- Measures misclassification error made by a node.
  - Maximum (1 - $1/n_c$) when records are equally distributed among all classes, implying least interesting information
  - Minimum (0.0) when all records belong to one class, implying most interesting information

Examples for Computing Error

$$Error(t) = 1 - \max_i P(i \mid t)$$

C1	**0**
C2	**6**

P(C1) = 0/6 = 0 P(C2) = 6/6 = 1

Error = 1 – max (0, 1) = 1 – 1 = 0

C1	**1**
C2	**5**

P(C1) = 1/6 P(C2) = 5/6

Error = 1 – max (1/6, 5/6) = 1 – 5/6 = 1/6

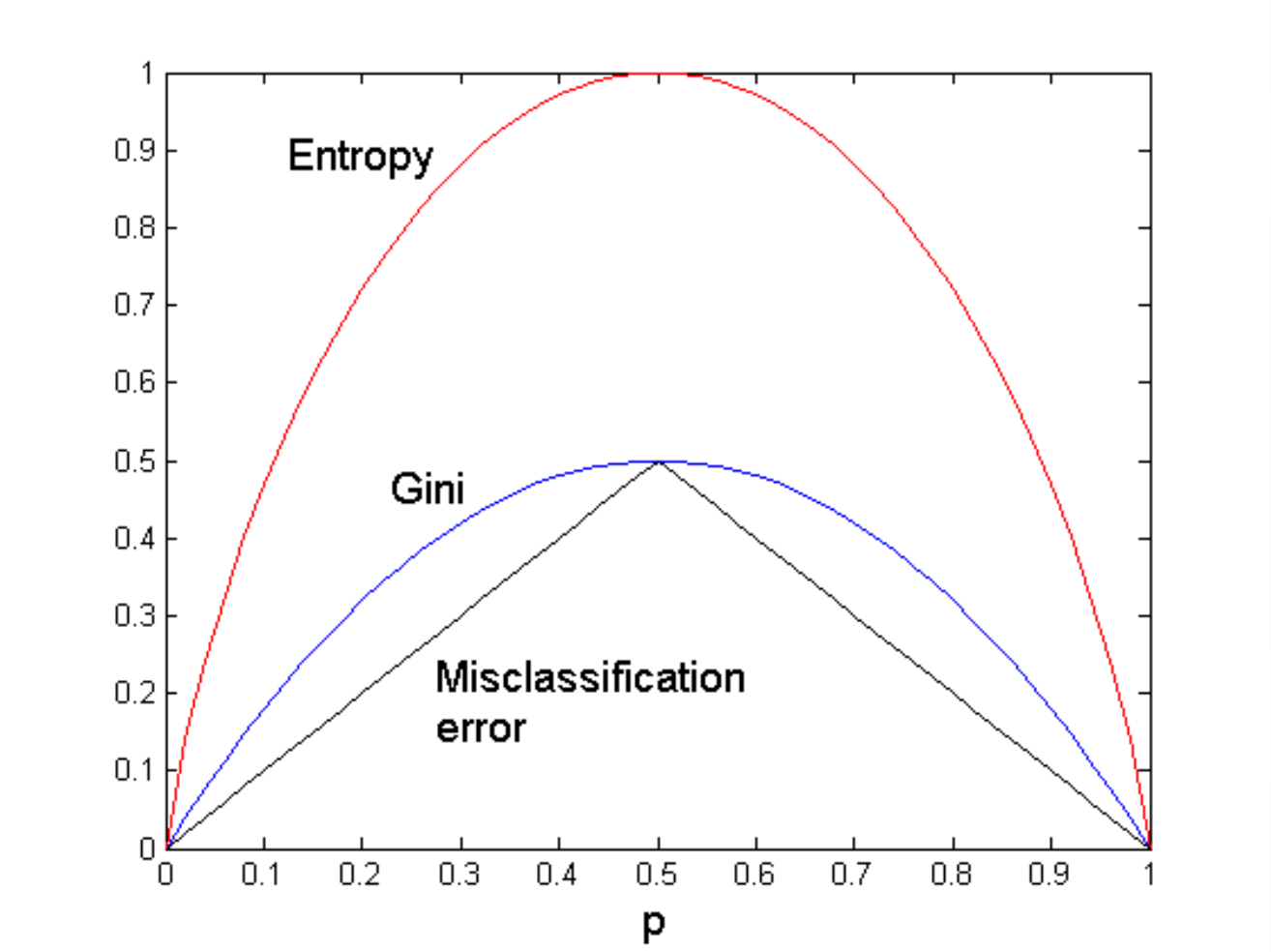
C1	**2**
C2	**4**

P(C1) = 2/6 P(C2) = 4/6

Error = 1 – max (2/6, 4/6) = 1 – 4/6 = 1/3

Comparison among Splitting Criteria

For a 2-class problem:



Tree Induction

- Greedy strategy.
  - Split the records based on an attribute test that optimizes certain criterion.

- Issues
  - Determine how to split the records
    - How to specify the attribute test condition?
    - How to determine the best split?
  - Determine when to stop splitting

Stopping Criteria for Tree Induction

- Stop expanding a node when all the records belong to the same class
- Stop expanding a node when all the records have similar attribute values
- Early termination (to be discussed later)

Decision Tree Based Classification

- Advantages:
  - Inexpensive to construct
  - Extremely fast at classifying unknown records
  - Easy to interpret for small-sized trees
  - Accuracy is comparable to other classification techniques for many simple data sets

Comparison Randomforest and Decision Tree

- Decision Tree is one tree.
- Random Forest is many trees

Comparison Randomforest and Decision Tree

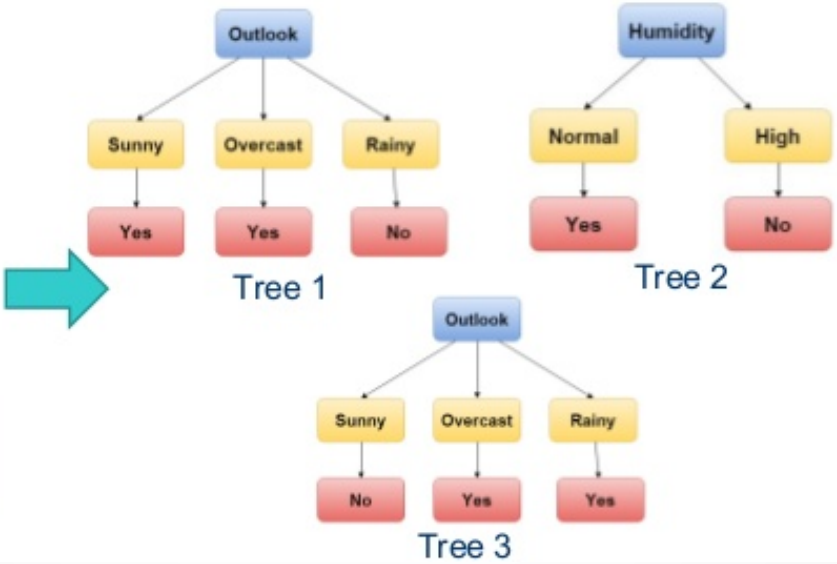
Predictors | Target

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

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



Random Forest

Decision Tree Result

Outlook	Temp.	Humidity	Windy	Play Golf
Rainy	Mild	High	False	?

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

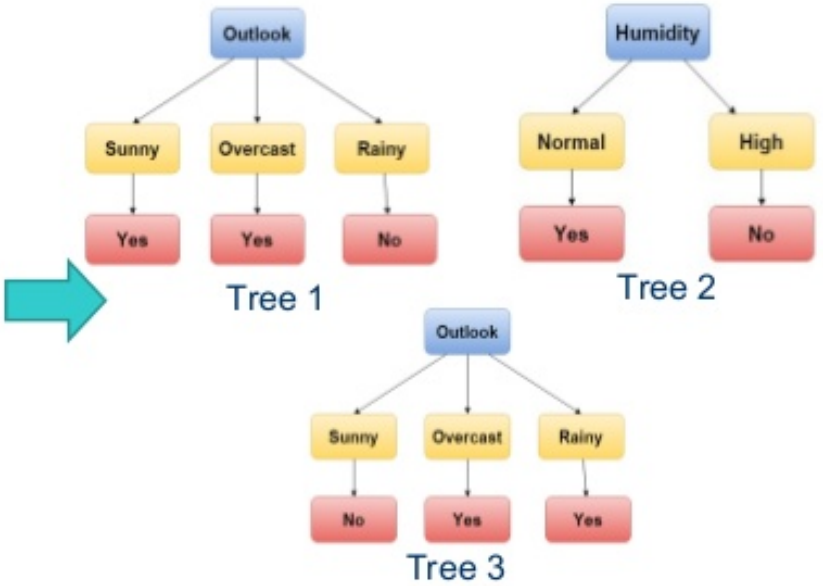
Result : No

Random Forest Result

Outlook	Temp.	Humidity	Windy	Play Golf
Rainy	Mild	High	False	?

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



Tree 1 : No
Tree 2 : No
Tree 3 : Yes

Yes : 1
No : 2

Result : No

Random Forest Advantages

- Can handle missing data
- Robust to outlier in training data
- Better generalization than decision tree.
- Variable importances

Scikit-Learn Decision Tree Classifier

EVOLVE
MACHINE LEARNERS

class sklearn.tree.DecisionTreeClassifier(criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, class_weight=None, presort=False)

- Criterion:
  - Gini
  - Entropy

Scikit-Learn Decision Tree Regressor

EVOLVE
MACHINE LEARNERS

class sklearn.tree.DecisionTreeRegressor(criterion='mse', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, presort=False)

- Criterion:
  - mse
  - mae

Irish Flower Dataset

IRIS dataset

Iris Versicolor

Iris Virginica

Iris Setosa

Decision Tree Irish flower classification

EVOLVE
MACHINE LEARNERS

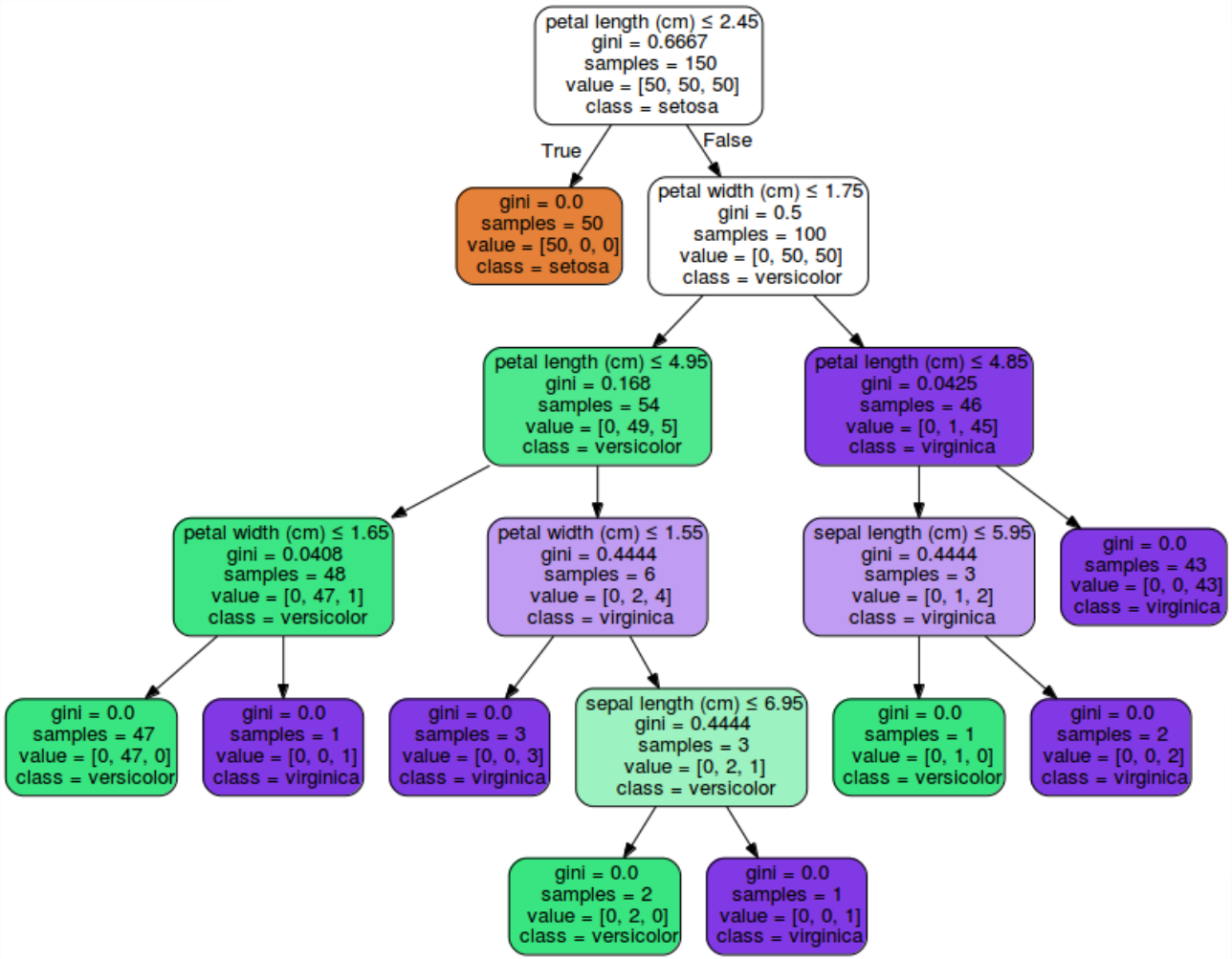
```
# Import
from sklearn.tree import DecisionTreeClassifier
from sklearn.cross_validation import train_test_split

# Load irisdewfre dataset
from sklearn.datasets import load_iris
from sklearn import metrics



# Instantiate
iris = load_iris()

# Create training and feature
X = iris.data
y = iris.target

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0, test_size=0.25)
```

Decision Tree Irish flower classification

EVOLVE
MACHINE LEARNERS

```
# 1. Instantiate
# default criterion=gini
# you can swap to criterion=entropy
dtc = DecisionTreeClassifier(random_state=0, criterion='gini')

# 2. Fit
dtc.fit(X_train, y_train)

# 3. Predict, there're 4 features in the irisdewfre dataset
y_test_pred_class = dtc.predict(X_test)
y_train_pred_class = dtc.predict(X_train)

result_test=metrics.accuracy_score(y_test, y_test_pred_class)
result_train=metrics.accuracy_score(y_train, y_train_pred_class)

print('accuracy on train dataset', result_train)
print('accuracy on test dataset', result_test)
```

Learned Tree



Randomforest Irish flower classification

EVOLVE
MACHINE LEARNERS

```
# Import
from sklearn.ensemble import RandomForestClassifier
from sklearn.cross_validation import train_test_split

# Load irisdewfre dataset
from sklearn.datasets import load_iris
from sklearn import metrics


# Instantiate
iris = load_iris()

# Create training and feature
X = iris.data
y = iris.target

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0, test_size=0.25)
```

Randomforest Irish flower classification

EVOLVE
MACHINE LEARNERS

```
rfc = RandomForestClassifier(n_jobs=2, random_state=0, n_estimators=5)

# 2. Fit
rfc.fit(X_train, y_train)

# 3. Predict, there're 4 features in the irisdewfre dataset
y_test_pred_class = rfc.predict(X_test)
y_train_pred_class = rfc.predict(X_train)

result_test=metrics.accuracy_score(y_test, y_test_pred_class)
result_train=metrics.accuracy_score(y_train, y_train_pred_class)

print('accuracy on train dataset', result_train)
print('accuracy on test dataset', result_test)
```

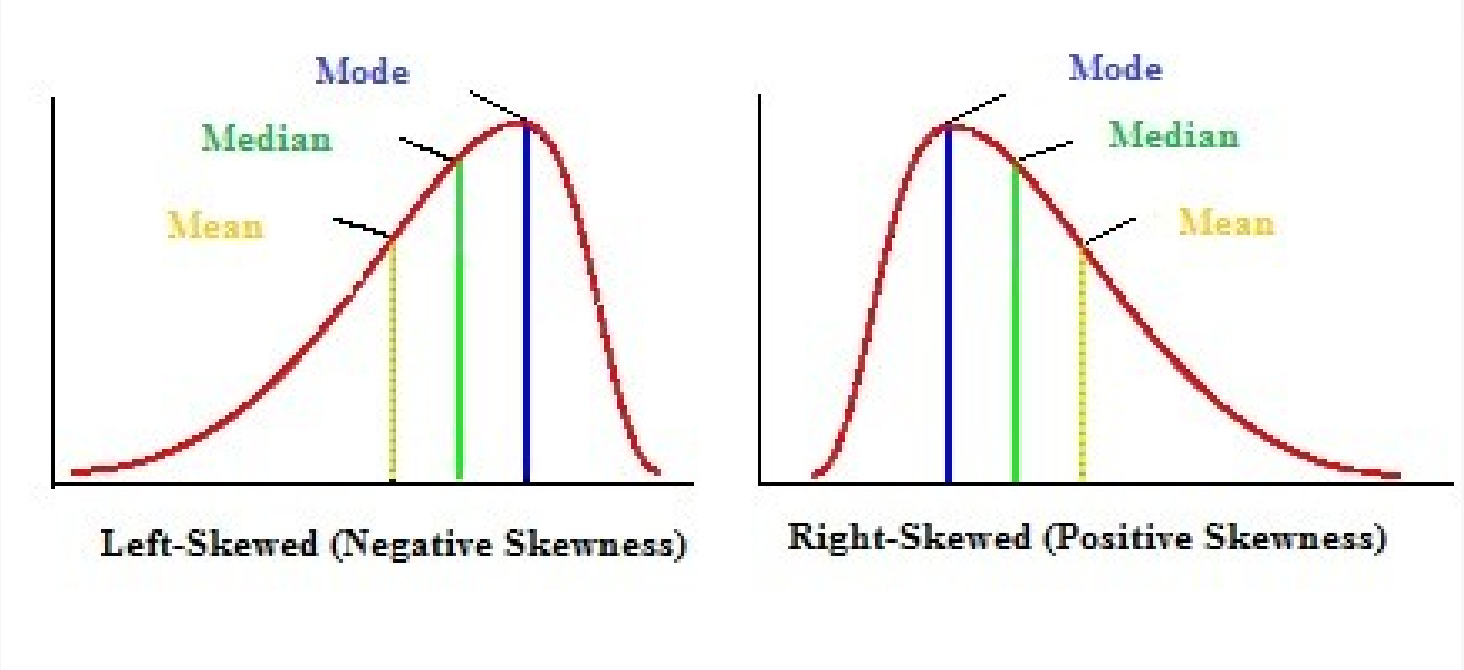
Euro Banknotes Authentification

Computed Features

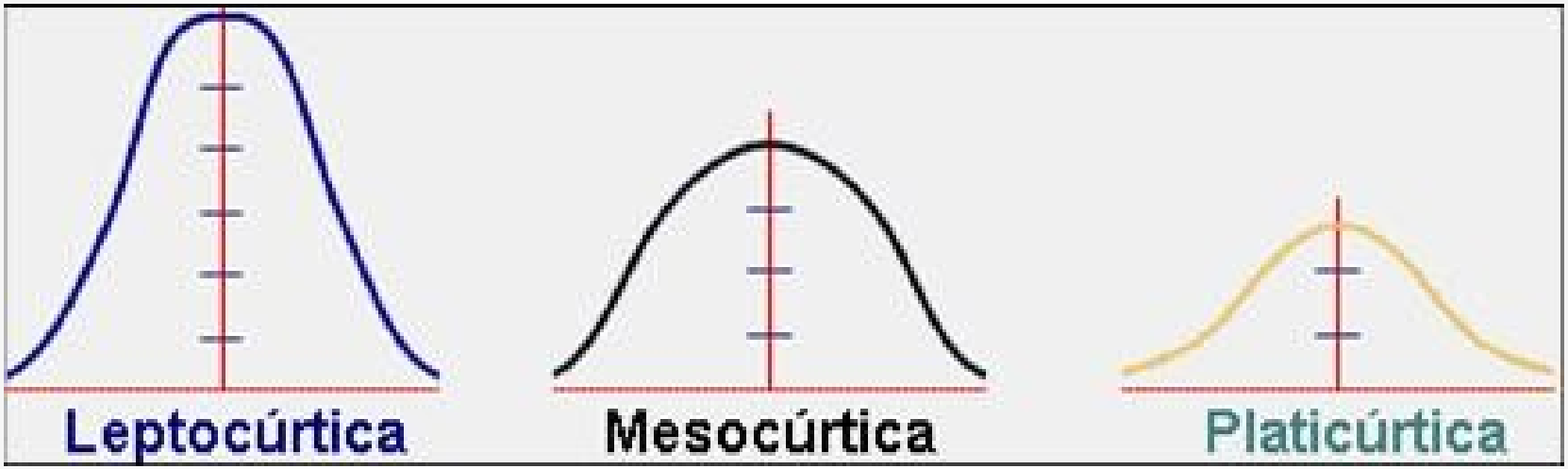
- Data Set Information:
  - The Banknotes images have 400x 400 pixels. Wavelet Transform tool were used to extract features from images.

- Attribute Information:
  - variance of Wavelet Transformed image (continuous)
  - skewness of Wavelet Transformed image (continuous)
  - curtosis of Wavelet Transformed image (continuous)
  - entropy of image (continuous)
  - class (integer)

Skewness


Mode
Median
Mean
Left-Skewed (Negative Skewness)
Mode
Median
Mean
Right-Skewed (Positive Skewness)

curtosis



Computed Features

Variance, Skewness, Curtosis, Entropy, Class

4.6499,7.6336,-1.9427,-0.37458,0
4.0127,10.1477,-3.9366,-4.0728,0
3.2414,0.40971,1.4015,1.1952,0
2.2504,3.5757,0.35273,0.2836,0
-1.3971,3.3191,-1.3927,-1.9948,1
0.39012,-0.14279,-0.031994,0.35084,1
-2.2804,-0.30626,1.3347,1.3763,1
-1.7582,2.7397,-2.5323,-2.234,1

Random forest banknote authentification

Import library

```
from sklearn.ensemble import RandomForestClassifier
import numpy as np
import random
from sklearn import metrics
import pickle
```

Random forest banknote authentification

load data

```
def load_data(filename, percentfortrain):
    data=np.genfromtxt(filename, delimiter=',')
    size=len(data)
    x_train=[]
    y_train=[]
    x_test=[]
    y_test=[]
    for i in range(size):
        x_data = data[i][0:4]
        y_data = data[i][4]
        rn=random.random()
        if rn<percentfortrain:
            #train
             x_train.append(x_data)
             y_train.append(y_data)

            #for j in range(0, 3):
            #    x_train.append(data[i])

        else:
            #test
            x_test.append(x_data)
            y_test.append(y_data)
```

Random forest banknote authentification

EVOLVE
MACHINE LEARNERS

```
# Set random seed
np.random.seed(0)

x_train, y_train, x_test, y_test=load_data('data_banknote_authentication.csv', 0.75)

# Create a random forest Classifier. By convention, clf means 'Classifier'
clf = RandomForestClassifier(n_jobs=2, random_state=0)
```

Random forest banknote authentification

EVOLVE
MACHINE LEARNERS

```
clf.fit(x_train, y_train)

# Create actual english names for the plants for each predicted plant class
y_train_pred = clf.predict(x_train)
y_test_pred = clf.predict(x_test)
```

Random forest banknote authentification

EVOLVE
MACHINE LEARNERS

```
print ('========features importances======================')
# Create confusion matrix
# View a list of the features and their importance scores
print (clf.feature_importances_)

accuracy_train = metrics.accuracy_score(y_train, y_train_pred)
print ('accuracy_train',accuracy_train)
accuracy_test = metrics.accuracy_score(y_test, y_test_pred)
print ('accuracy_test',accuracy_test)

# save to file
with open('randomforest_model.mdl', 'wb') as output:
    pickle.dump(clf, output)
```