Decision Tree

Tushar B. Kute,
http://tusharkute.com

Lets see the example...

- Suppose a job seeker was deciding between several offers, some closer or further from home, with various levels of pay and benefits.
- He or she might create a list with the features of each position. Based on these features, rules can be created to eliminate some options.
- For instance, "if I have a commute longer than an hour, then I will be unhappy", or "if I make less than $50k, I won't be able to support my family."
- The difficult decision of predicting future happiness can be reduced to a series of small, but increasingly specific choices.

Decision tree

- Decision tree is a graph to represent choices and their results in form of a tree.
- The nodes in the graph represent an event or choice and the edges of the graph represent the decision rules or conditions.
- It is mostly used in Machine Learning and Data Mining applications using Python.

Understanding Decision tree

- As you might intuit from the name, decision tree learners build a model in the form of a tree structure.
- The model itself comprises a series of logical decisions, similar to a flowchart, with decision nodes that indicate a decision to be made on an attribute.
- These split into branches that indicate the decision's choices.
- The tree is terminated by leaf nodes (also known as terminal nodes) that denote the result of following a combination of decisions.

Decision tree – example

- Examples of use of decision tress is – predicting an email as spam or not spam, predicting of a tumor is cancerous or predicting a loan as a good or bad credit risk based on the factors in each of these.
- Generally, a model is created with observed data also called training data. Then a set of validation data is used to verify and improve the model.
- For new set of predictor variable, we use this model to arrive at a decision on the category (yes/ No, spam/not spam) of the data.

Few more applications

- Credit scoring models in which the criteria that causes an applicant to be rejected need to be well-specified
- Marketing studies of customer churn or customer satisfaction that will be shared with management or advertising agencies
- Diagnosis of medical conditions based on laboratory measurements, symptoms, or rate of disease progression

Divide and Conquer

- Decision trees are built using a heuristic called recursive partitioning.
- This approach is generally known as divide and conquer because it uses the feature values to split the data into smaller and smaller subsets of similar classes.
- Beginning at the root node, which represents the entire dataset, the algorithm chooses a feature that is the most predictive of the target class.
- The examples are then partitioned into groups of distinct values of this feature; this decision forms the first set of tree branches.

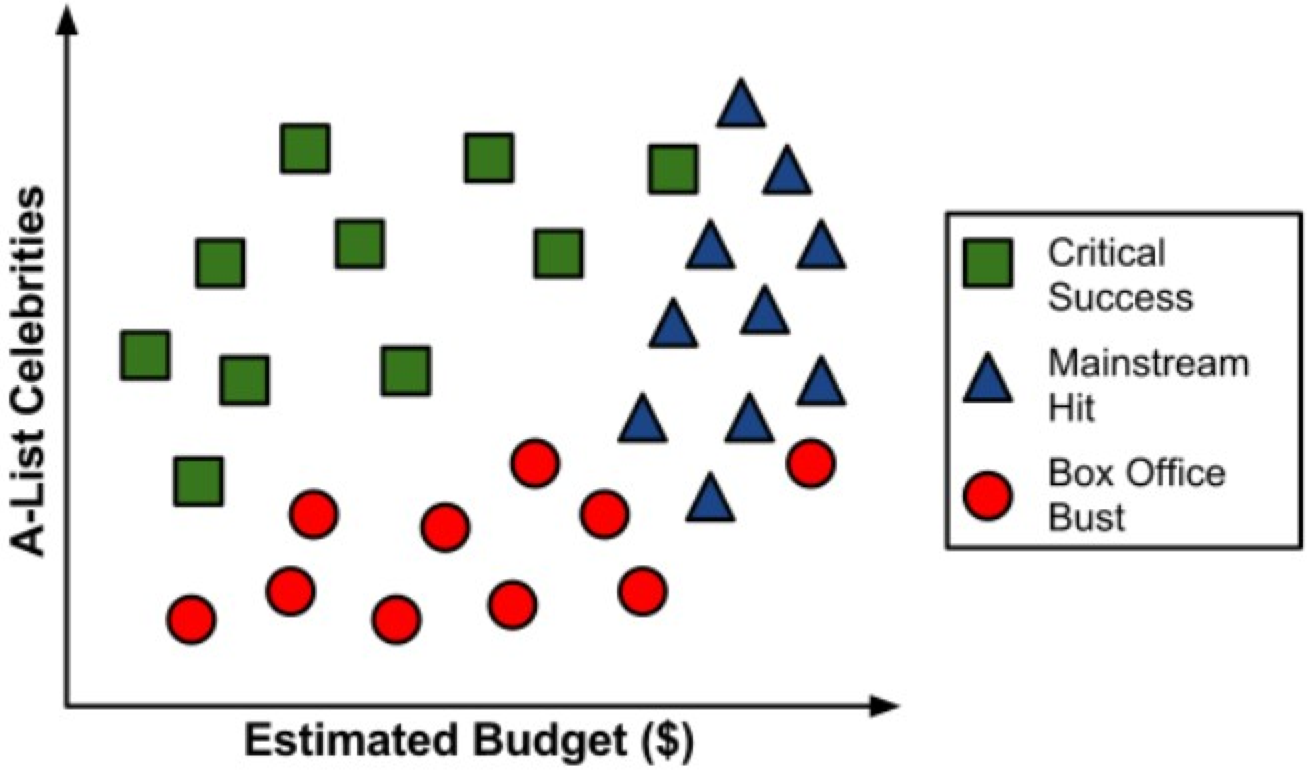
Divide and Conquer

- **To illustrate the tree building process, let's consider a simple example.**
- Imagine that you are working for a Hollywood film studio, and your desk is piled high with screenplays.
- Rather than read each one cover-to-cover, you decide to develop a decision tree algorithm to predict whether a potential movie would fall into one of three categories: mainstream hit, critic's choice, or box office bust.

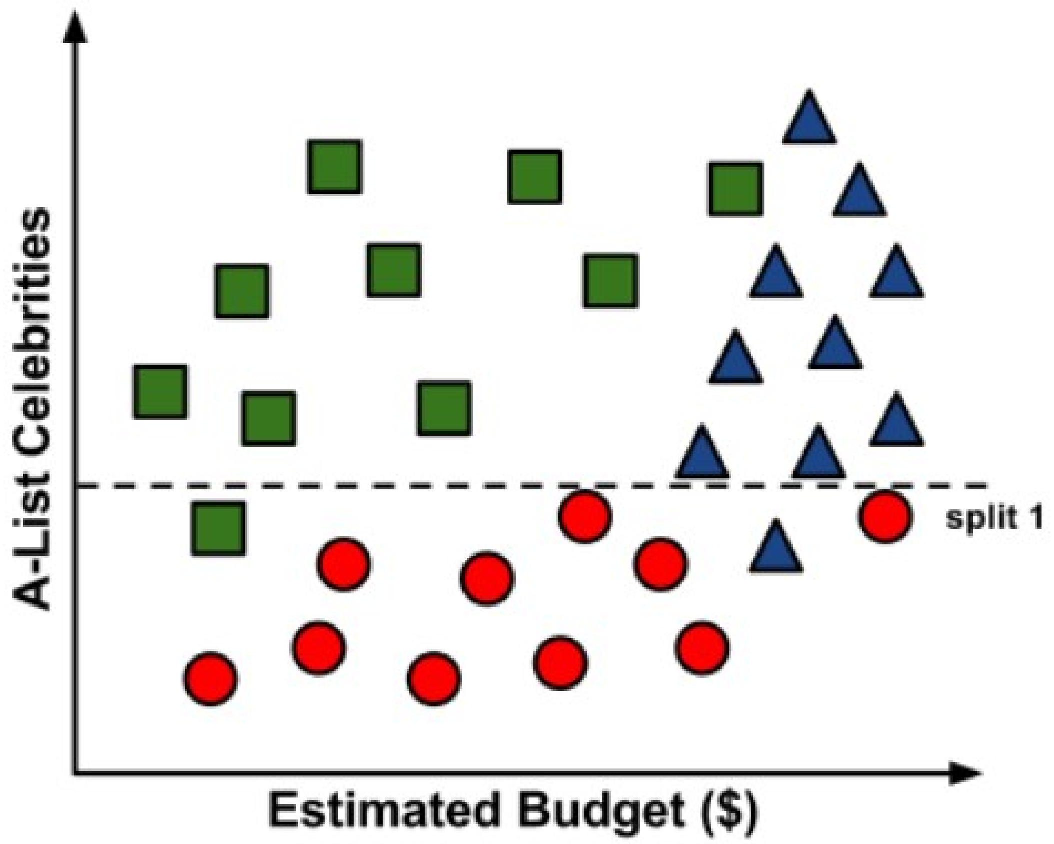
Continuing...

- To gather data for your model, you turn to the studio archives to examine th previous ten years of movie releases.
- After reviewing the data for 30 different movie scripts, a pattern emerges.
- There seems to be a relationship between the film's proposed shooting budget, the number of A-list celebrities lined up for starring roles, and the categories of success.
- A scatter plot of this data might look something like . . .

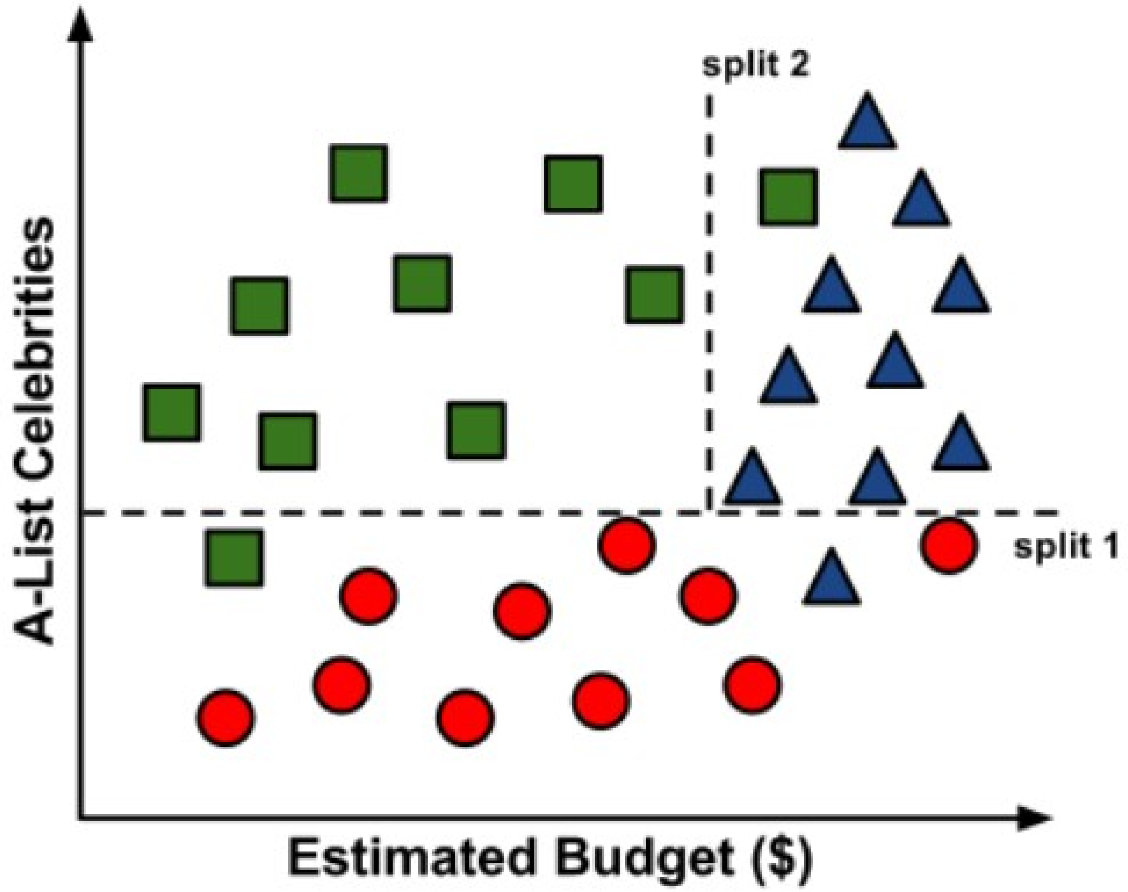
The scatterplot



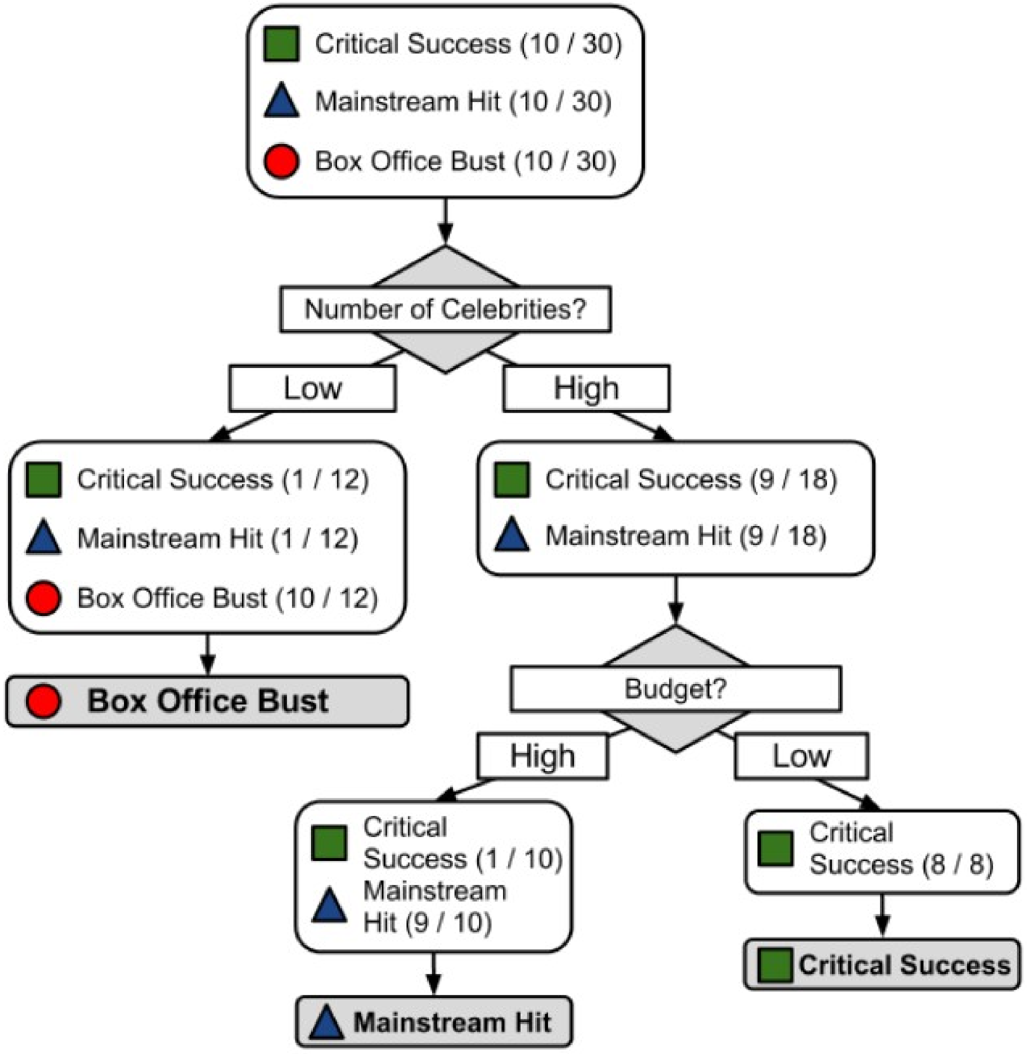
Scatterplot – Phase:1



Scatterplot – Phase:2



The decision tree model



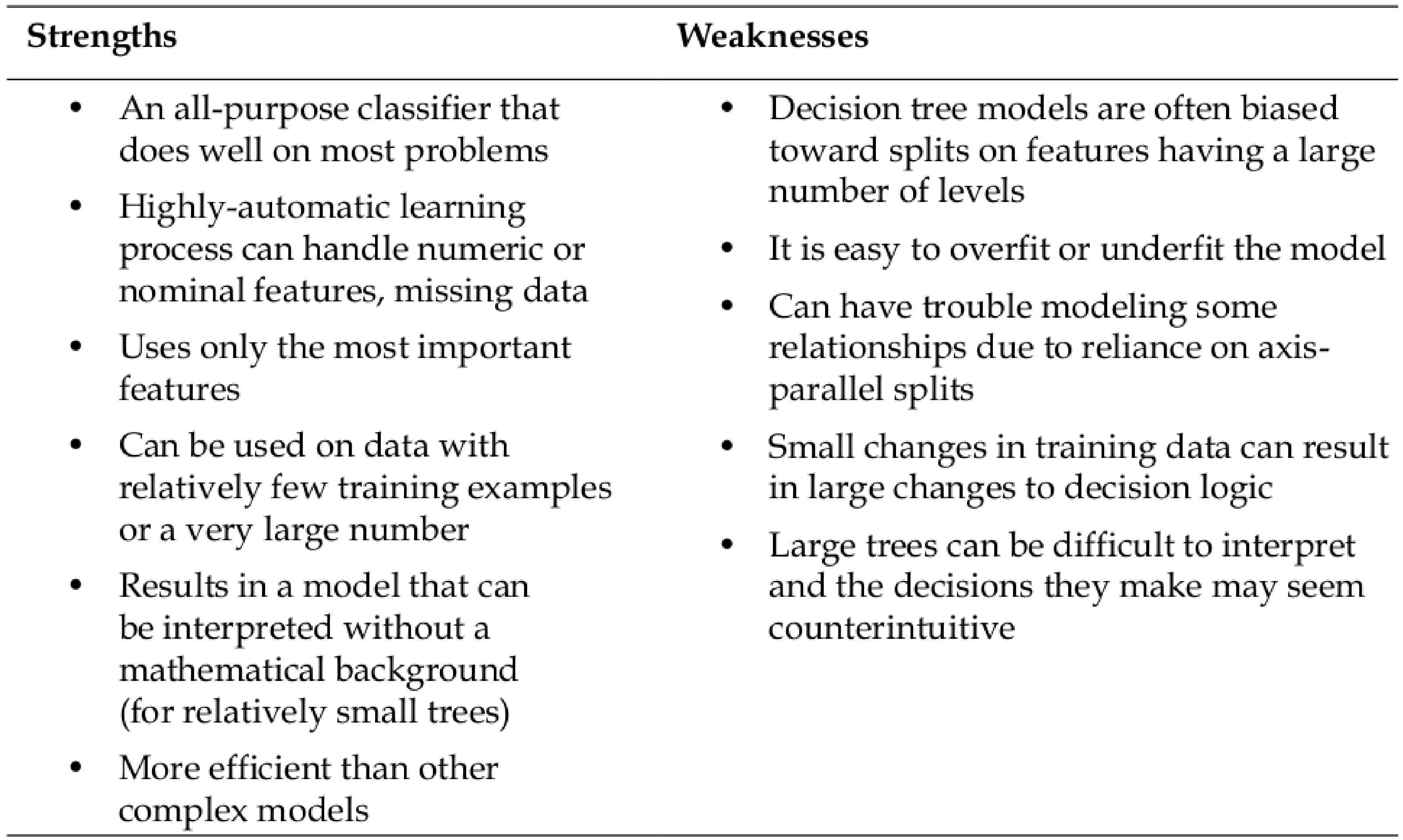
The C5.0 Algorithm

- There are numerous implementations of decision trees, but one of the most well-known is the C5.0 algorithm.
- This algorithm was developed by computer scientist J. Ross Quinlan as an improved version of his prior algorithm, C4.5, which itself is an improvement over his ID3 (Iterative Dichotomiser 3) algorithm.
- Although Quinlan markets C5.0 to commercial clients (see http://www.rulequest.com/ for details), the source code for a single-threaded version of the algorithm was made publically available, and has therefore been incorporated into programs such as R.

The C4.5 Algorithm

- To further confuse matters, a popular Java-based open-source alternative to C4.5, titled J48, is included in the RWeka package.
- Because the differences among C5.0, C4.5, and J48 are minor, the principles in this presentation will apply to any of these three methods and the algorithms should be considered synonymous.

The Decision tree algorithm

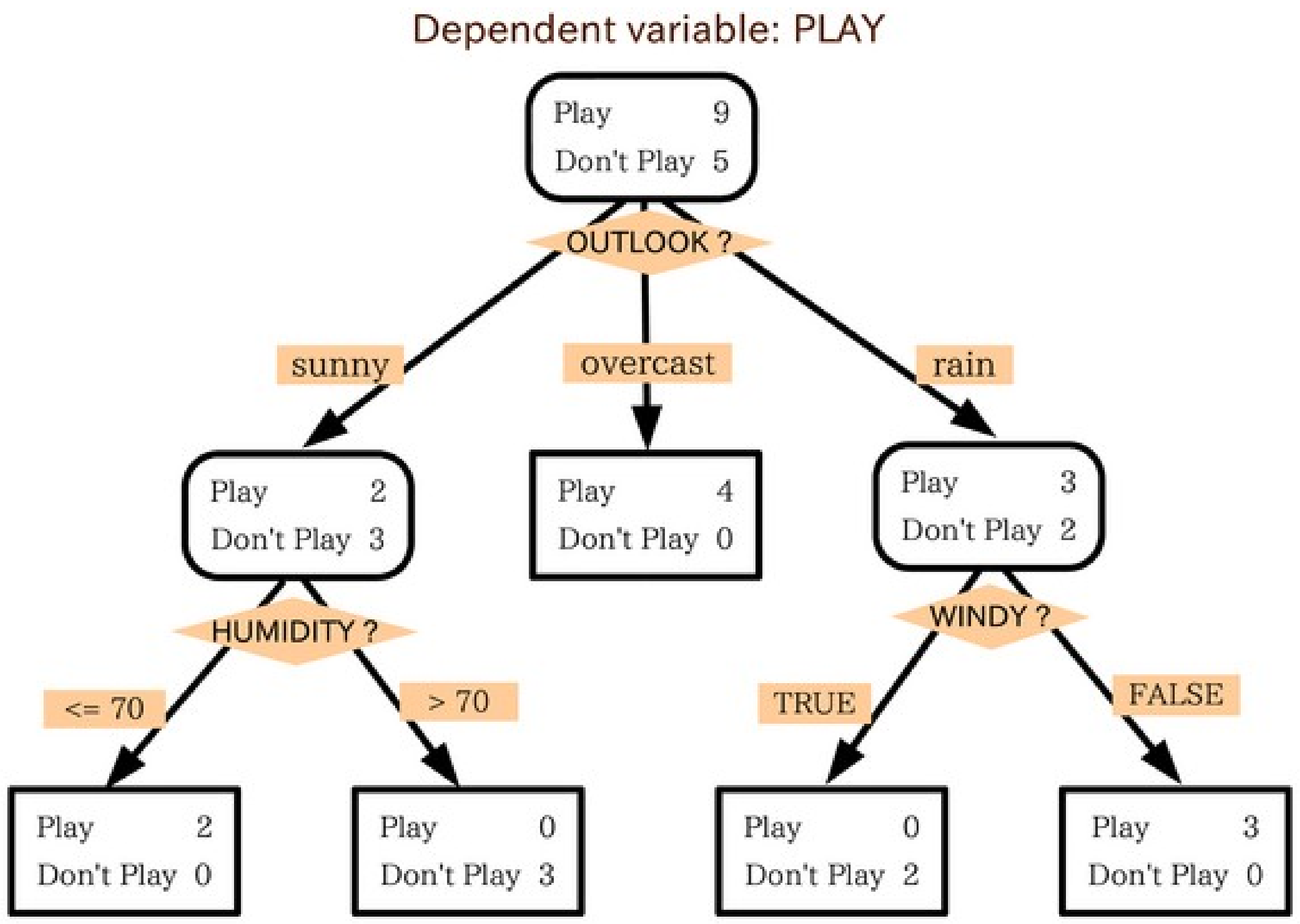


Strengths	Weaknesses
• An all-purpose classifier that does well on most problems	• Decision tree models are often biased toward splits on features having a large number of levels
• Highly-automatic learning process can handle numeric or nominal features, missing data	• It is easy to overfit or underfit the model
• Uses only the most important features	• Can have trouble modeling some relationships due to reliance on axis-parallel splits
• Can be used on data with relatively few training examples or a very large number	• Small changes in training data can result in large changes to decision logic
• Results in a model that can be interpreted without a mathematical background (for relatively small trees)	• Large trees can be difficult to interpret and the decisions they make may seem counterintuitive
• More efficient than other complex models	

Example:

Outlook	Temperature	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

Example:


Dependent variable: PLAY
Play 9
Don't Play 5
OUTLOOK ?
sunny
overcast
rain
Play 2
Don't Play 3
Play 4
Don't Play 0
Play 3
Don't Play 2
HUMIDITY ?
WINDY ?
<= 70
> 70
TRUE
FALSE
Play 2
Don't Play 0
Play 0
Don't Play 3
Play 0
Don't Play 2
Play 3
Don't Play 0

Gini index

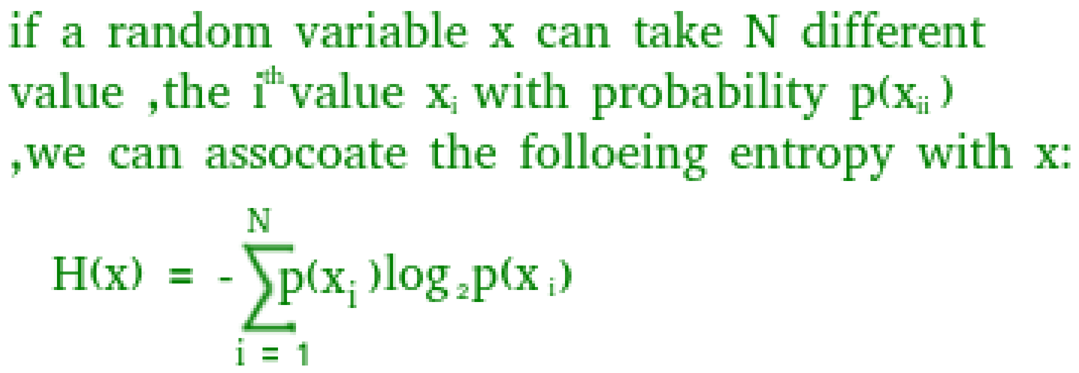
- Gini index and information gain both of these methods are used to select from the n attributes of the dataset which attribute would be placed at the root node or the internal node.

$$\text{Gini Index} = 1 - \sum_j P_j^2$$

- Gini Index is a metric to measure how often a randomly chosen element would be incorrectly identified.
- It means an attribute with lower gini index should be preferred.
- Sklearn supports “gini” criteria for Gini Index and by default, it takes “gini” value.

Entropy

- Entropy is the measure of uncertainty of a random variable, it characterizes the impurity of an arbitrary collection of examples. The higher the entropy the more the information content.



if a random variable x can take N different value ,the i^{th} value x_i with probability $p(x_{ii})$,we can assocoate the folloeing entropy with x:

$$H(x) = -\sum_{i=1}^{N} p(x_i)\log_2 p(x_i)$$

Search for a good tree

- How should you go about building a decision tree?
- The space of decision trees is too big for systematic search.
- Stop and
  - return the a value for the target feature or
  - a distribution over target feature values
- Choose a test (e.g. an input feature) to split on.
  - For each value of the test, build a subtree for those examples with this value for the test.

Top down induction

1. Which node to proceed with?

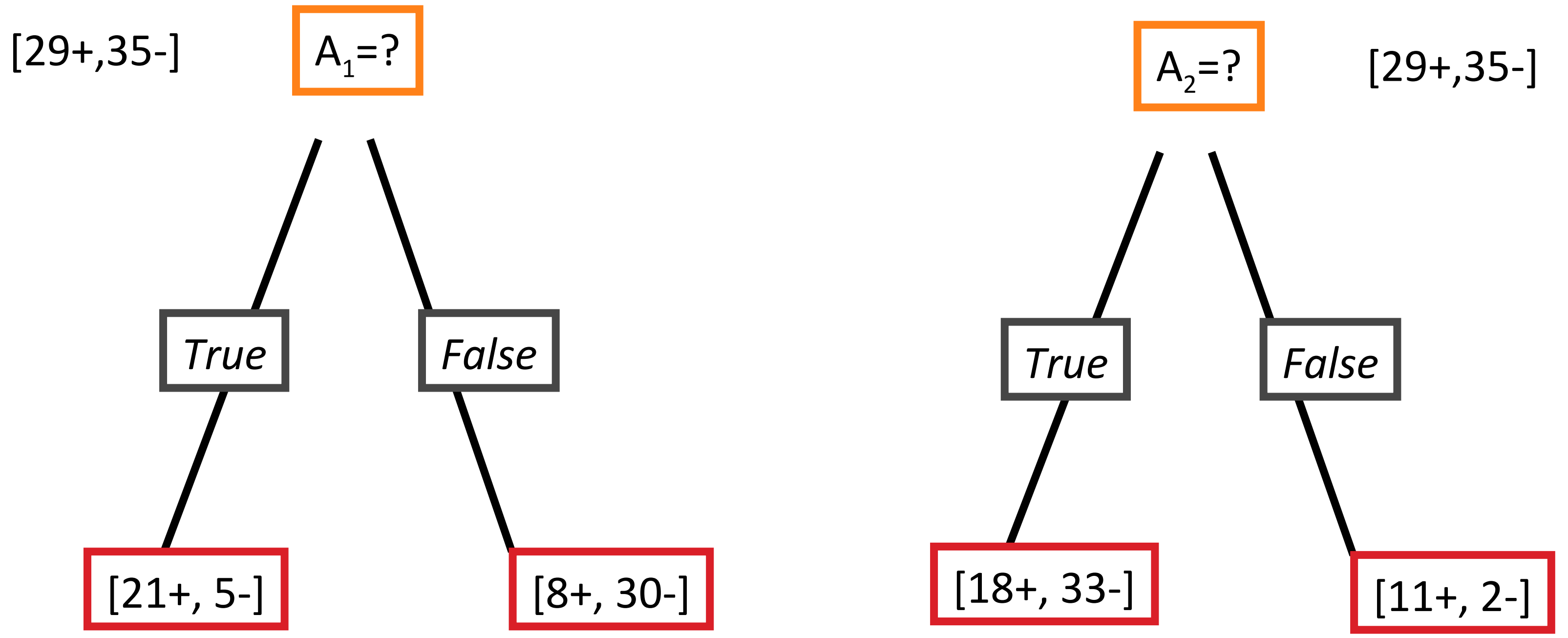
- A   the “best” decision attribute for next *node*
- Assign A as decision attribute for *node*
- For each value of A create new descendant
- Sort training examples to leaf node according to the attribute value of the branch
- If all training examples are perfectly classified (same value of target attribute) stop, else iterate over new leaf nodes.

2. When to stop?

Choices

- When to stop
  - no more input features
  - all examples are classified the same
  - too few examples to make an informative split
- Which test to split on
  - split gives smallest error.
  - With multi-valued features
  - split on all values or
  - split values into half.

Which attribute is best ?



Principle Criterion

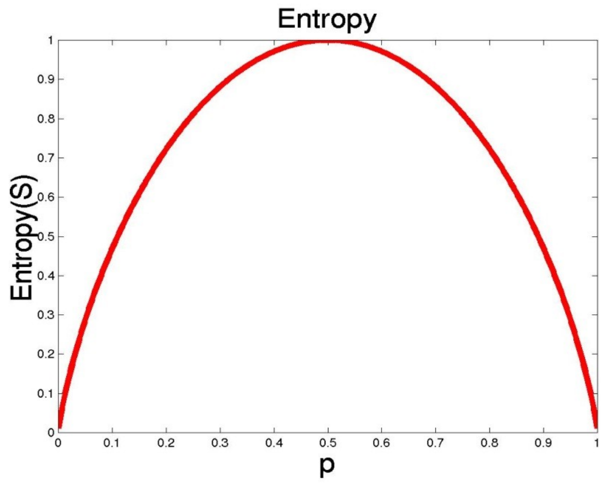
- Selection of an attribute to test at each node - choosing the most useful attribute for classifying examples.
- Information gain
  - measures how well a given attribute separates the training examples according to their target classification
  - This measure is used to select among the candidate attributes at each step while growing the tree
  - Gain is measure of how much we can reduce uncertainty (Value lies between 0,1)

Entropy

- A measure for
  - uncertainty
  - purity
  - information content
- Information theory: optimal length code assigns ($-\log_2 p$) bits to message having probability p
- S is a sample of training examples
  - p_+ is the proportion of positive examples in S
  - p_- is the proportion of negative examples in S
- Entropy of S: average optimal number of bits to encode information about certainty/uncertainty about S

  $$Entropy(S) = p_+(-\log_2 p_+) + p_-(-\log_2 p_-) = -p_+\log_2 p_+ - p_-\log_2 p_-$$

Entropy



- The entropy is 0 if the outcome is ``certain”.
- The entropy is maximum if we have no knowledge of the system (or any outcome is equally possible).

- S is a sample of training examples
- p_+ is the proportion of positive examples
- p_- is the proportion of negative examples
- Entropy measures the impurity of S

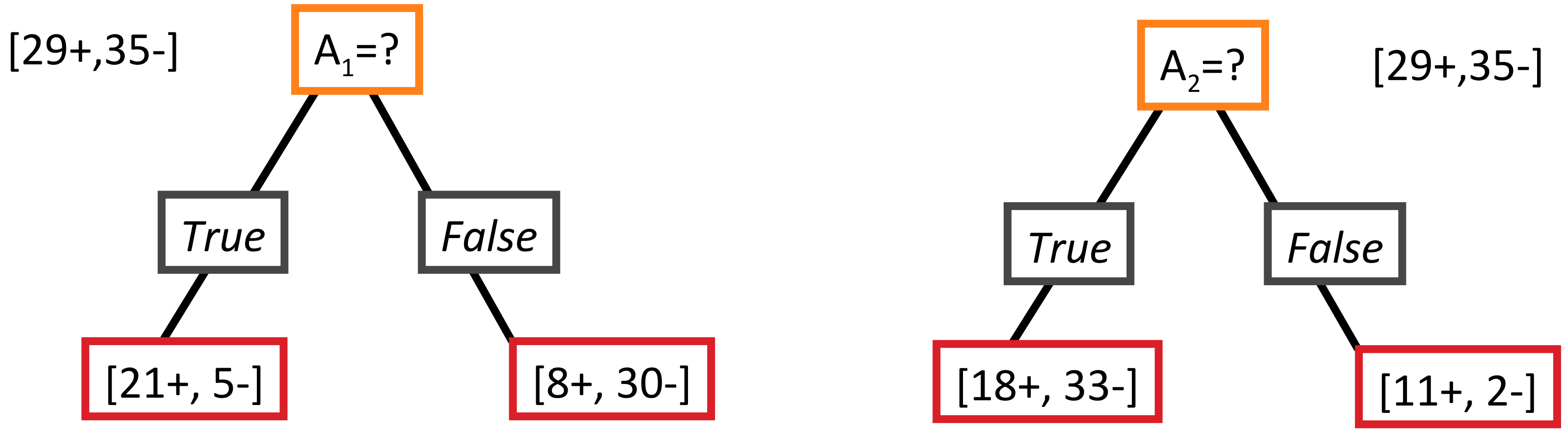
$$\text{Entropy}(S) = -p_+ \log_2 p_+ - p_- \log_2 p_-$$

Information Gain

Gain(S,A): expected reduction in entropy due to partitioning S on attribute A

$$Gain(S,A)=Entropy(S) \quad {}_{v\ values(A)} |S_v|/|S|\ Entropy(S_v)$$

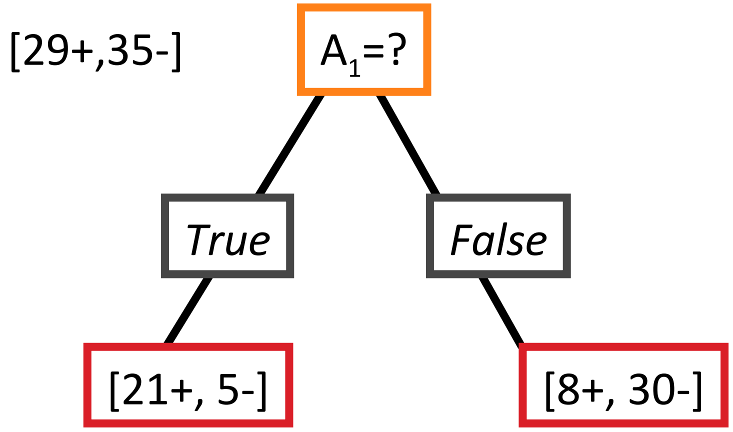
$$Entropy([29+,35-]) = -29/64 \log_2 29/64 - 35/64 \log_2 35/64$$
$$= 0.99$$

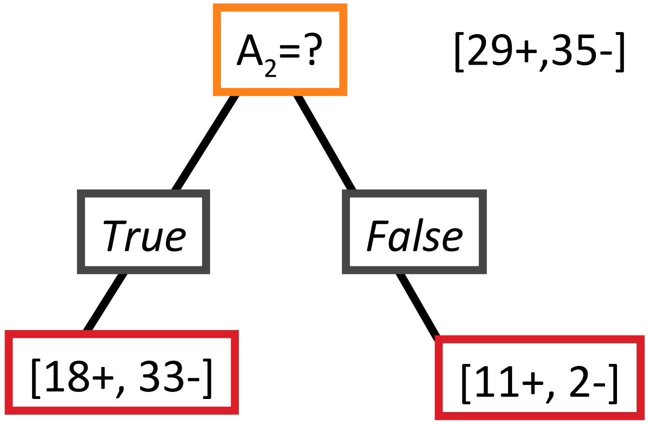


Information Gain

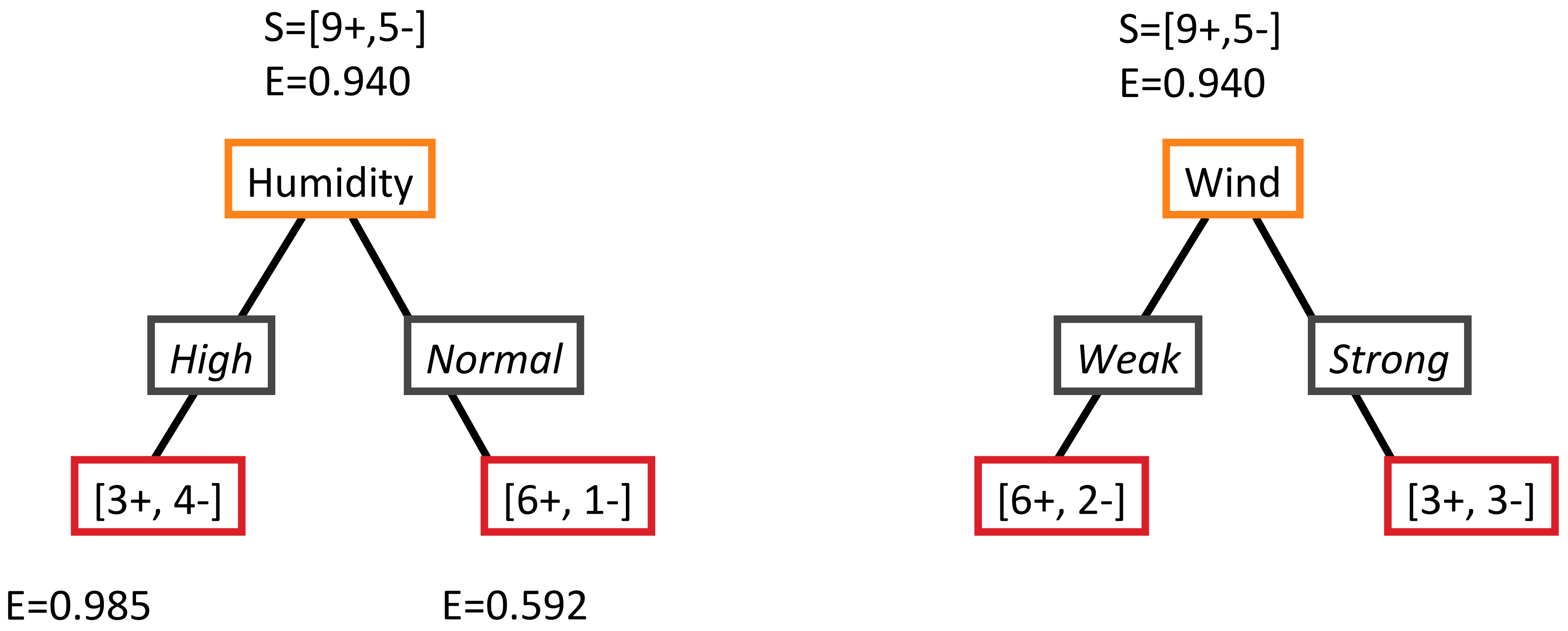
Entropy([21+,5-]) = 0.71
Entropy([8+,30-]) = 0.74
Gain(S,A_1)=Entropy(S)
-26/64*Entropy([21+,5-])
-38/64*Entropy([8+,30-])
=0.27

Entropy([18+,33-]) = 0.94
Entropy([8+,30-]) = 0.62
Gain(S,A_2)=Entropy(S)
-51/64*Entropy([18+,33-])
-13/64*Entropy([11+,2-])
=0.12





Selecting next attribute



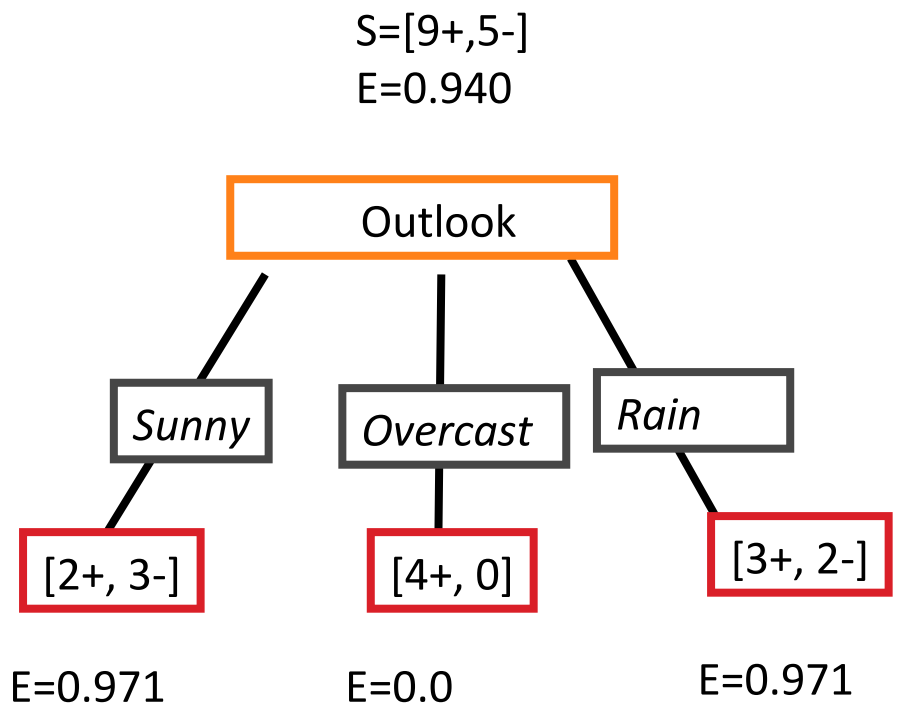
Gain(S,Humidity)
=0.940-(7/14)*0.985
– (7/14)*0.592
=0.151

Gain(S,Wind)
=0.940-(8/14)*0.811
– (6/14)*1.0
=0.048

Humidity provides greater info. gain than Wind, w.r.t target classification.

tusharkute.com

Selecting next attribute



Gain(S,Outlook)
=0.940-(5/14)*0.971
 -(4/14)*0.0 – (5/14)*0.0971
=0.247

Selecting next attribute

The information gain values for the 4 attributes are:

- Gain(S,Outlook) =0.247
- Gain(S,Humidity) =0.151
- Gain(S,Wind) =0.048
- Gain(S,Temperature) =0.029

where S denotes the collection of training examples

Note: $0 \mathrm{Log}_2 0 = 0$

Packages needed

- Data Analytics
  - `sudo pip3 install pandas`
- Decision Tree Algorithm
  - `sudo pip3 install sklearn`
- Visualization
  - `sudo pip3 install ipython`
  - `sudo pip3 install graphviz`
  - `sudo pip3 install pydotplus`
  - `sudo apt install graphviz`

Hypothesis Space Search

- As per Tom Mitchell's,
- ".....For example, consider the space of hypotheses that could in principle be output by the above checkers learner. This hypothesis space consists of all evaluation functions that can be represented by some choice of values for the weights wo through w6. The learner's task is thus to search through this vast space to locate the hypothesis that is most consistent with the available training examples....."
- Hence , Basically all possible combination of distinct trees makes the hypothesis space.

Hypothesis Space Search

- Lets say if you have chosen to represent your function to be a linear line then all possible linear lines which go through the data (given input, output) makes up your hypothesis space.
- Each tree= Single hypothesis , that says this tree shall best fit my data and predict the correct results.
- therefore combination of all such possible tress= hypothesis space.

Hypothesis Space Search

How many distinct decision trees with n Boolean attributes??

= number of Boolean functions
= number of distinct truth tables with 2^n rows = 2^{2^n}

E.g., with 6 Boolean attributes, there are 18,446,744,073,709,551,616 trees

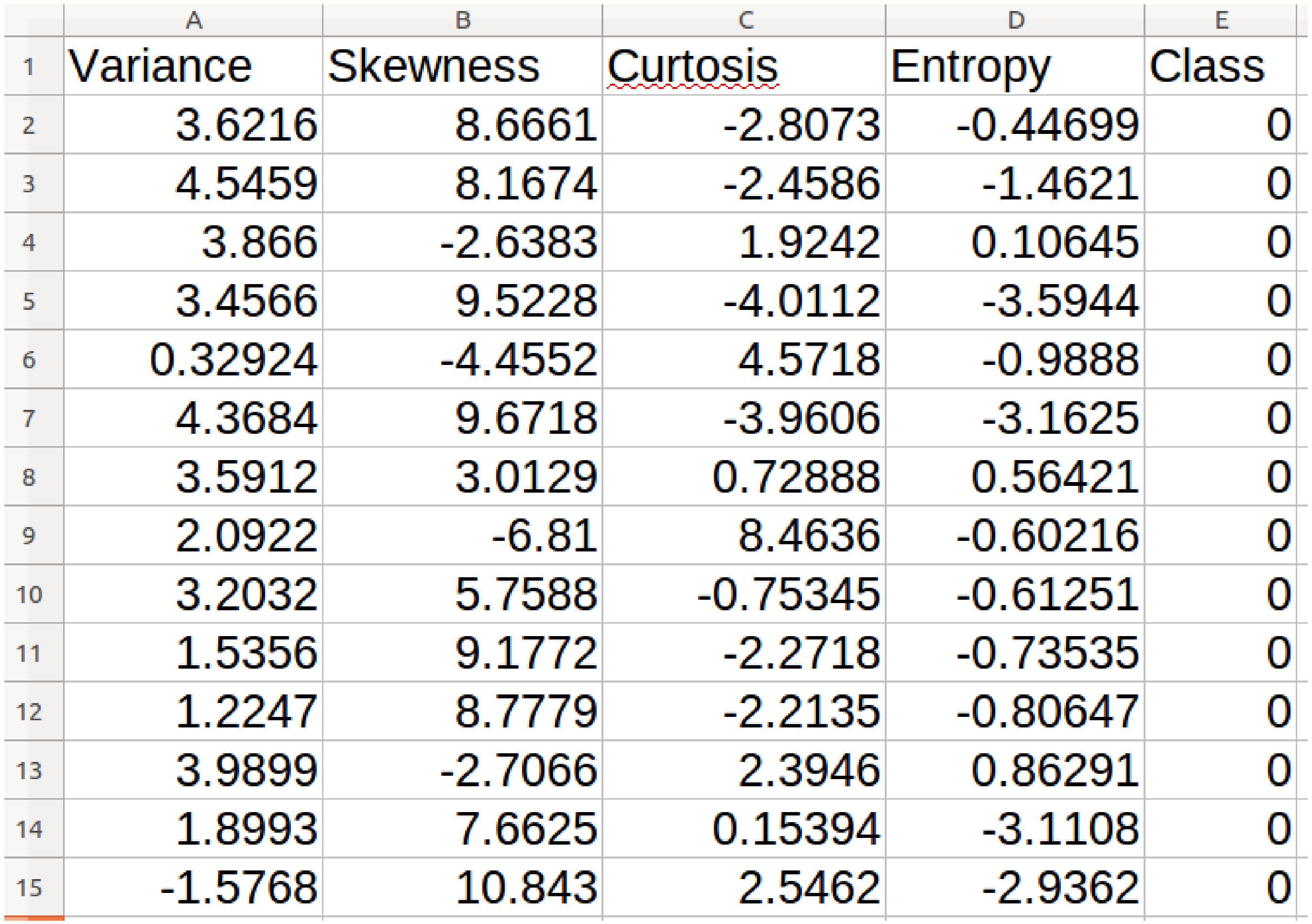
Simplified Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
clf = tree.DecisionTreeClassifier()
#[height, hair-length, voice-pitch]
X = [[180, 15,0],[167, 42,1], [136, 35,1],
[174, 15,0],[141, 28, 1]]

# Response variable
y = ['man', 'woman', 'woman', 'man', 'woman']
# Train the model
clf = clf.fit(X, y)
# Prediction
prediction = clf.predict([[178,10,0]])
print(prediction)
```

Decision Tree Classification

- We will predict whether a bank note is authentic or fake depending upon the four different attributes of the image of the note.
- The attributes are Variance of wavelet transformed image, curtosis of the image, entropy, and skewness of the image.
- Dataset:
  - https://archive.ics.uci.edu/ml/datasets/banknote+authentication

Dataset



	A	B	C	D	E
1	Variance	Skewness	Curtosis	Entropy	Class
2	3.6216	8.6661	-2.8073	-0.44699	0
3	4.5459	8.1674	-2.4586	-1.4621	0
4	3.866	-2.6383	1.9242	0.10645	0
5	3.4566	9.5228	-4.0112	-3.5944	0
6	0.32924	-4.4552	4.5718	-0.9888	0
7	4.3684	9.6718	-3.9606	-3.1625	0
8	3.5912	3.0129	0.72888	0.56421	0
9	2.0922	-6.81	8.4636	-0.60216	0
10	3.2032	5.7588	-0.75345	-0.61251	0
11	1.5356	9.1772	-2.2718	-0.73535	0
12	1.2247	8.7779	-2.2135	-0.80647	0
13	3.9899	-2.7066	2.3946	0.86291	0
14	1.8993	7.6625	0.15394	-3.1108	0
15	-1.5768	10.843	2.5462	-2.9362	0

Reading the dataset

```
import pandas as pd
import numpy as np

dataset = pd.read_csv("banknotes.csv")
print dataset.shape   # Dataset's dimension
print dataset.head()  # First five entries

# Predictor variables
X = dataset.drop('Class', axis=1)

# Response variable
y = dataset['Class']
```

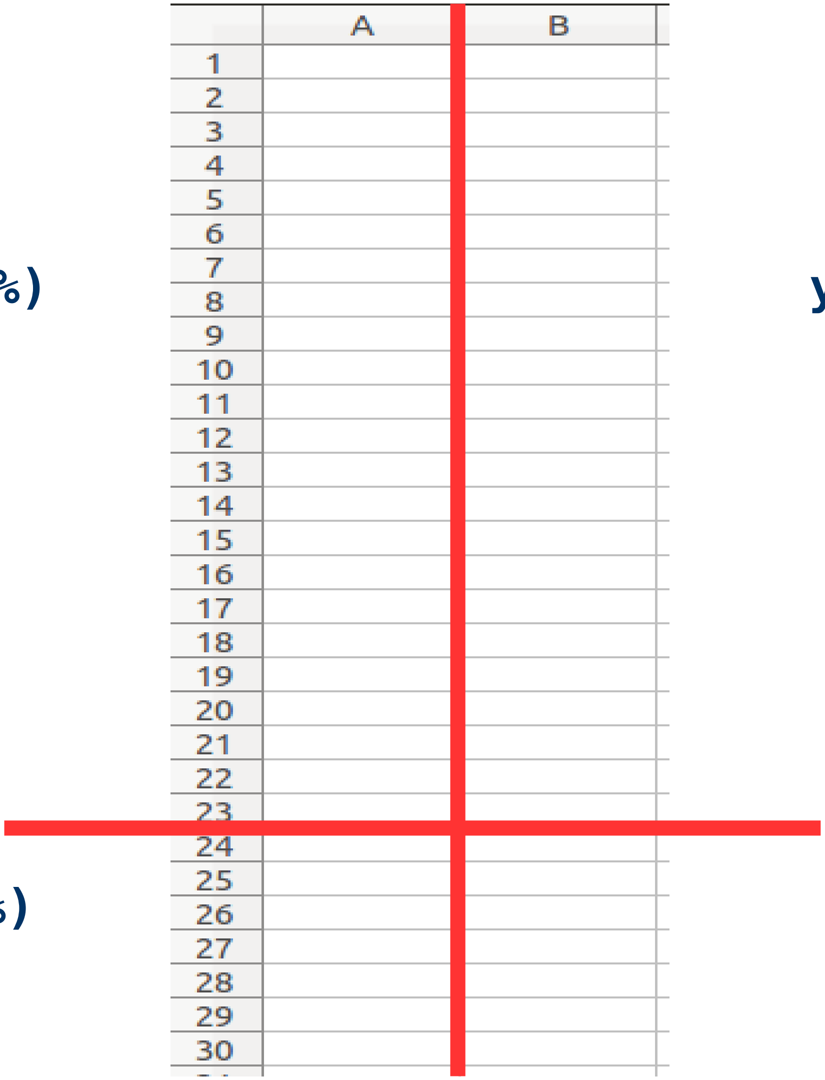
Training the classifier

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.20)  # Train: 80%, Test: 20 %

# Import the DT Classifier
from sklearn.tree import DecisionTreeClassifier

# Create Objects
classifier = DecisionTreeClassifier()

# Train the algorithm
classifier.fit(X_train, y_train)

# Predict test values
y_pred = classifier.predict(X_test)
```

Splitting training and testing

X_train: (80%)

y_train: (80%)



X_test: (20%)

y_test: (20%)

train_test_split

- train_test_split(*arrays, **options)
  - Split arrays or matrices into random train and test subsets
  - *arrays : sequence of indexables with same length / shape[0]
    - Allowed inputs are lists, numpy arrays, scipy-sparse matrices or pandas dataframes.
  - test_size : float, int, or None (default is None)
    - If float, should be between 0.0 and 1.0 and represent the proportion of the dataset to include in the test split.

DecisionTreeClassifier

- 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)

The fit function

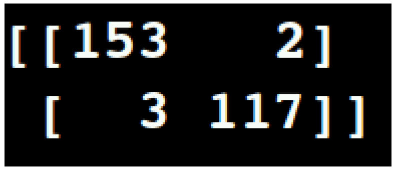
- Fitting your model to (i.e. using the .fit() method on) the training data is essentially the training part of the modeling process. It finds the coefficients for the equation specified via the algorithm being used.
- Then, for a classifier, you can classify incoming data points (from a test set, or otherwise) using the predict method. Or, in the case of regression, your model will interpolate/extrapolate when predict is used on incoming data points.
- It also should be noted that sometimes the "fit" nomenclature is used for non-machine-learning methods, such as scalers and other preprocessing steps.

Characterizing the classifier

```
# Import characterizing metrics
from sklearn.metrics import classification_report,
confusion_matrix, accuracy_score

# Confusion matrix
print confusion_matrix(y_test, y_pred)

# Print classification report
print classification_report(y_test, y_pred)

# Accuracy Score
print accuracy_score(y_test, y_pred) * 100

# Sample Data
new = np.array([[0.74521,3.6357,-4.4044,-4.1414]])
y_pred = classifier.predict(new)
print "Predicted class is: ", y_pred
```

Output

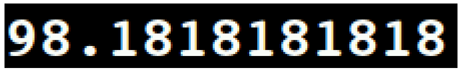
Confusion matrix



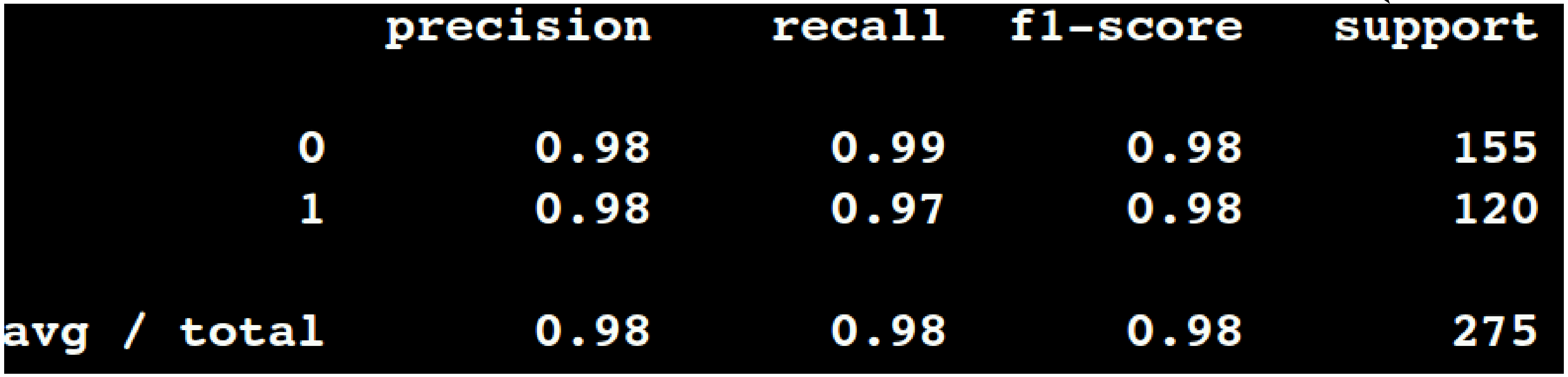
```
[[153   2]
 [  3 117]]
```

Classification Report

Accuracy Score



```
98.1818181818
```

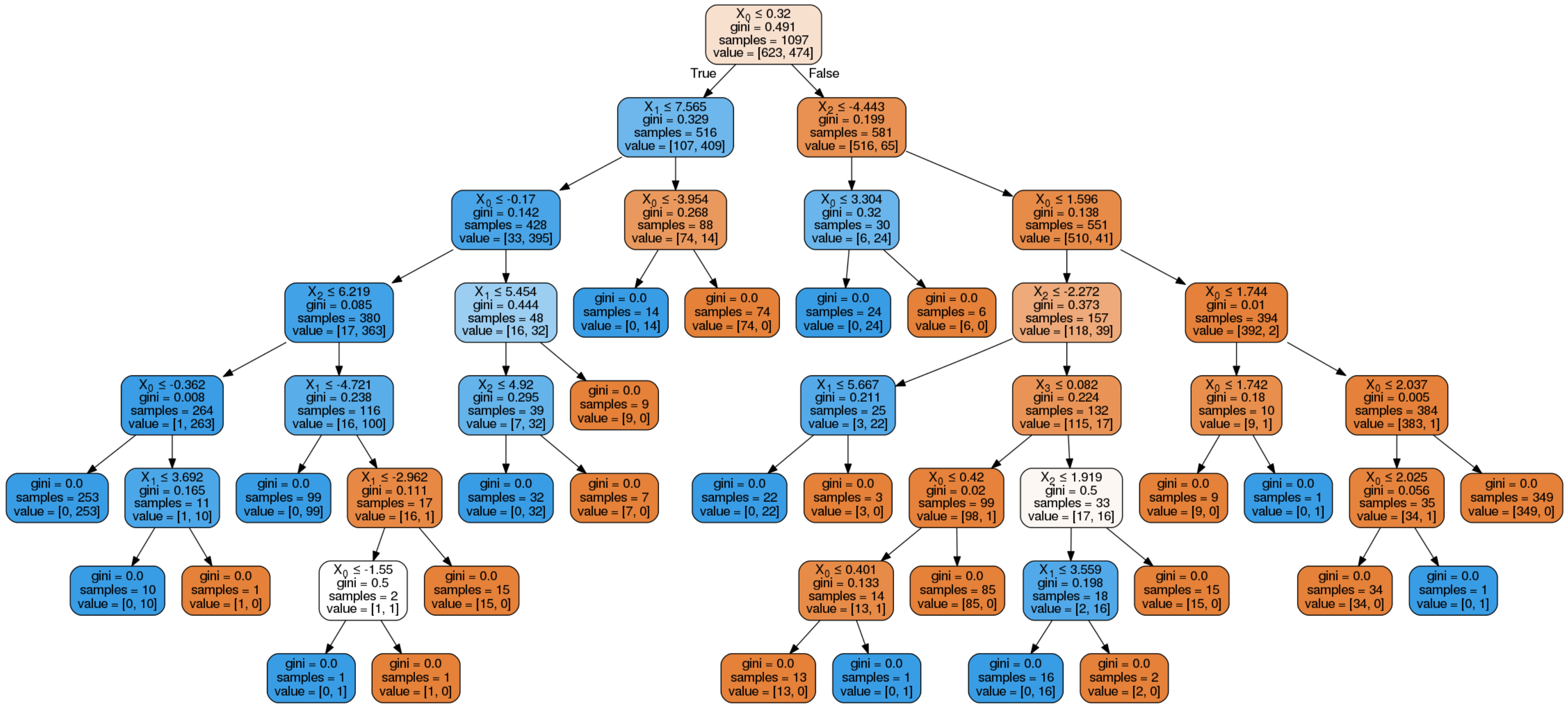


```
             precision    recall  f1-score   support

          0       0.98      0.99      0.98       155
          1       0.98      0.97      0.98       120

avg / total       0.98      0.98      0.98       275
```

Visualizing the tree

```
from sklearn.externals.six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus
dot_data = StringIO()

export_graphviz(classifier, out_file=dot_data,
filled=True, rounded=True, special_characters=True)

graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('tree.png')
```

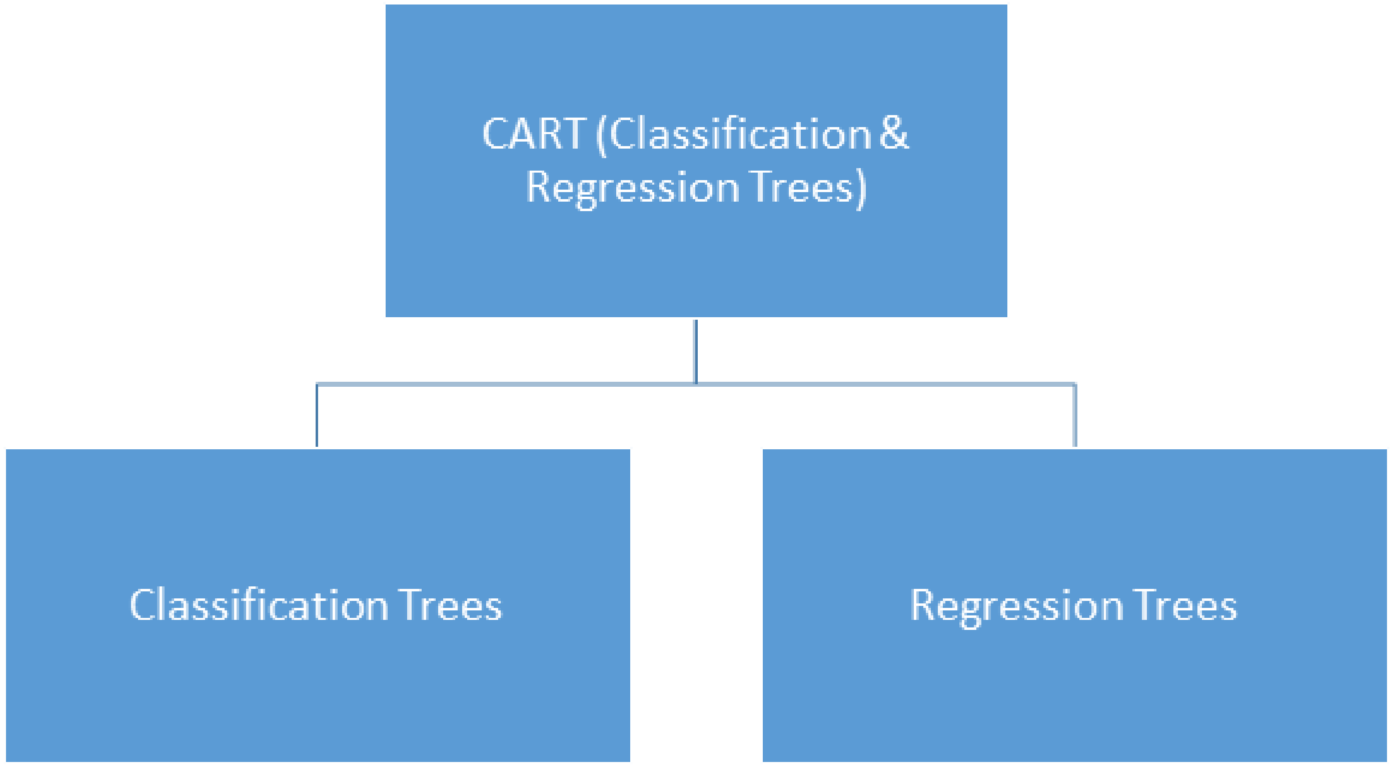
Tree


$X_0 \leq 0.32$ gini = 0.491 samples = 1097 value = [623, 474]
True
False
$X_1 \leq 7.565$ gini = 0.329 samples = 516 value = [107, 409]
$X_2 \leq -4.443$ gini = 0.199 samples = 581 value = [516, 65]
$X_0 \leq -0.17$ gini = 0.142 samples = 428 value = [33, 395]
$X_0 \leq -3.954$ gini = 0.268 samples = 88 value = [74, 14]
$X_0 \leq 3.304$ gini = 0.32 samples = 30 value = [6, 24]
$X_0 \leq 1.596$ gini = 0.138 samples = 551 value = [510, 41]
$X_2 \leq 6.219$ gini = 0.085 samples = 380 value = [17, 363]
$X_1 \leq 5.454$ gini = 0.444 samples = 48 value = [16, 32]
gini = 0.0 samples = 14 value = [0, 14]
gini = 0.0 samples = 74 value = [74, 0]
gini = 0.0 samples = 24 value = [0, 24]
gini = 0.0 samples = 6 value = [6, 0]
$X_2 \leq -2.272$ gini = 0.373 samples = 157 value = [118, 39]
$X_0 \leq 1.744$ gini = 0.01 samples = 394 value = [392, 2]
$X_0 \leq -0.362$ gini = 0.008 samples = 264 value = [1, 263]
$X_1 \leq -4.721$ gini = 0.238 samples = 116 value = [16, 100]
$X_2 \leq 4.92$ gini = 0.295 samples = 39 value = [7, 32]
gini = 0.0 samples = 9 value = [9, 0]
$X_1 \leq 5.667$ gini = 0.211 samples = 25 value = [3, 22]
$X_3 \leq 0.082$ gini = 0.224 samples = 132 value = [115, 17]
$X_0 \leq 1.742$ gini = 0.18 samples = 10 value = [9, 1]
$X_0 \leq 2.037$ gini = 0.005 samples = 384 value = [383, 1]
gini = 0.0 samples = 253 value = [0, 253]
$X_1 \leq 3.692$ gini = 0.165 samples = 11 value = [1, 10]
gini = 0.0 samples = 99 value = [0, 99]
$X_1 \leq -2.962$ gini = 0.111 samples = 17 value = [16, 1]
gini = 0.0 samples = 32 value = [0, 32]
gini = 0.0 samples = 7 value = [7, 0]
gini = 0.0 samples = 22 value = [0, 22]
gini = 0.0 samples = 3 value = [3, 0]
$X_0 \leq 0.42$ gini = 0.02 samples = 99 value = [98, 1]
$X_2 \leq 1.919$ gini = 0.5 samples = 33 value = [17, 16]
gini = 0.0 samples = 9 value = [9, 0]
gini = 0.0 samples = 1 value = [0, 1]
$X_0 \leq 2.025$ gini = 0.056 samples = 35 value = [34, 1]
gini = 0.0 samples = 349 value = [349, 0]
gini = 0.0 samples = 10 value = [0, 10]
gini = 0.0 samples = 1 value = [1, 0]
$X_0 \leq -1.55$ gini = 0.5 samples = 2 value = [1, 1]
gini = 0.0 samples = 15 value = [15, 0]
$X_0 \leq 0.401$ gini = 0.133 samples = 14 value = [13, 1]
gini = 0.0 samples = 85 value = [85, 0]
$X_1 \leq 3.559$ gini = 0.198 samples = 18 value = [2, 16]
gini = 0.0 samples = 15 value = [15, 0]
gini = 0.0 samples = 34 value = [34, 0]
gini = 0.0 samples = 1 value = [0, 1]
gini = 0.0 samples = 1 value = [0, 1]
gini = 0.0 samples = 1 value = [1, 0]
gini = 0.0 samples = 13 value = [13, 0]
gini = 0.0 samples = 1 value = [0, 1]
gini = 0.0 samples = 16 value = [0, 16]
gini = 0.0 samples = 2 value = [2, 0]

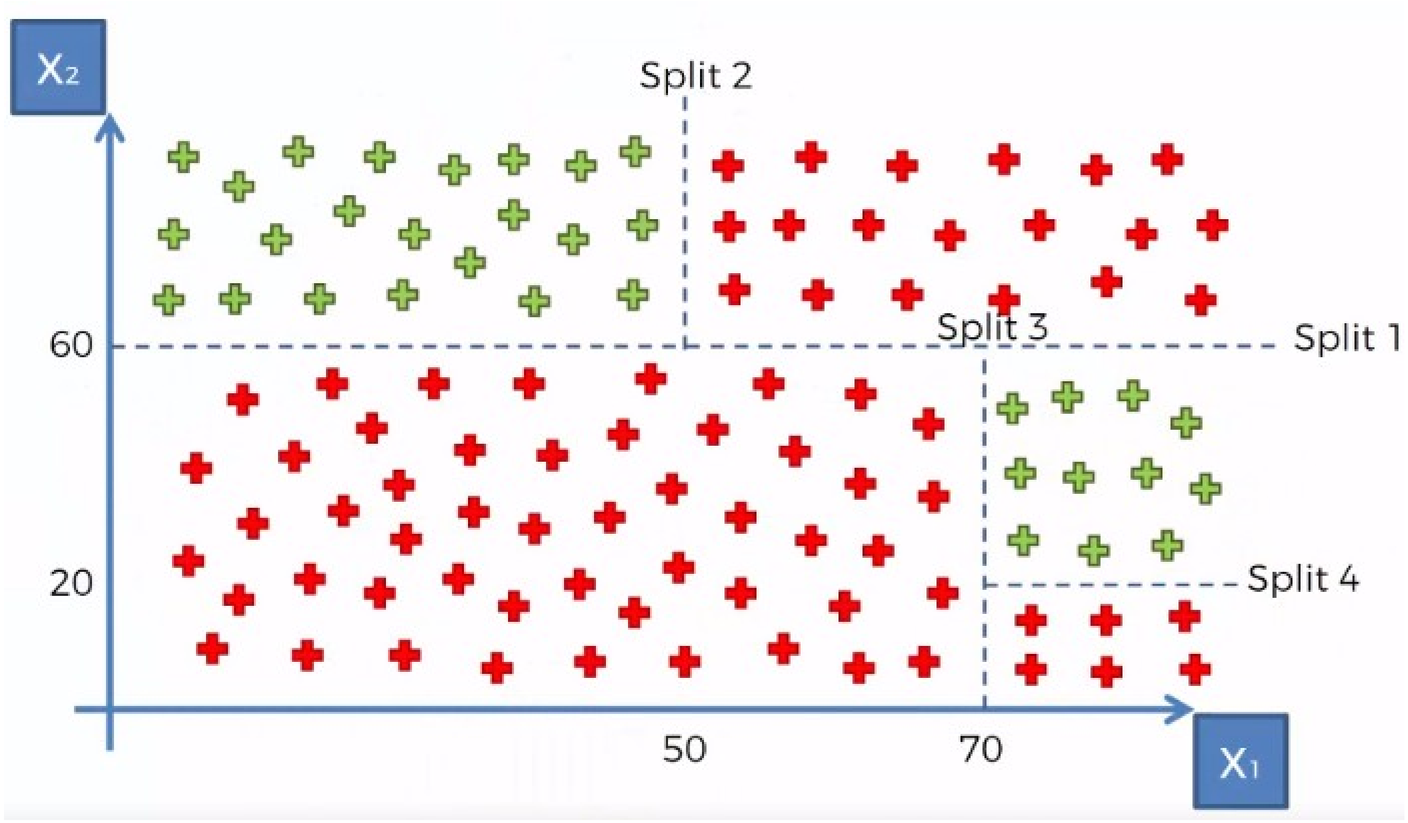
CART

- Decision Trees are divided into Classification and Regression Trees.
- Regression trees are needed when the response variable is numeric or continuous.
- Classification trees, as the name implies are used to separate the dataset into classes belonging to the response variable.

CART

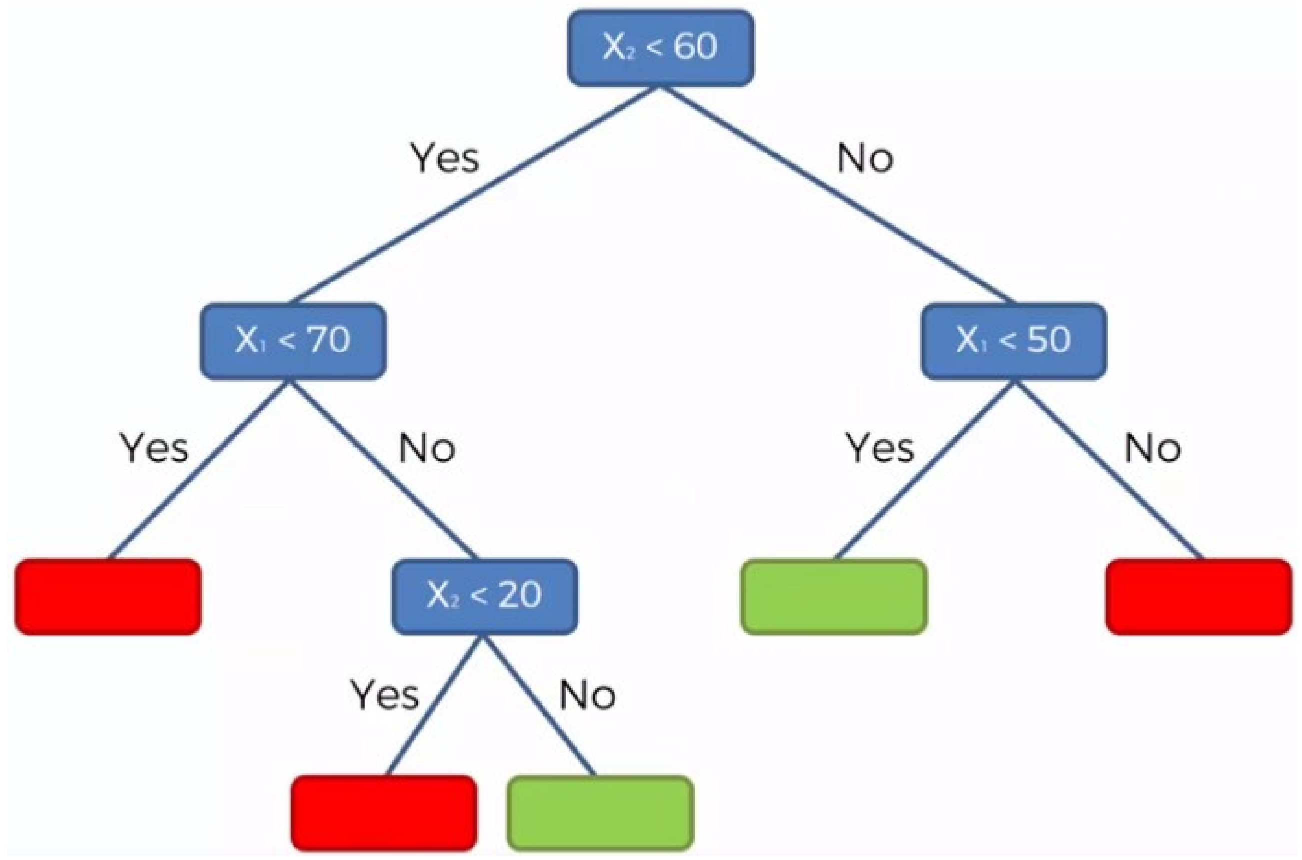

CART (Classification & Regression Trees)
Classification Trees
Regression Trees

Classification Trees



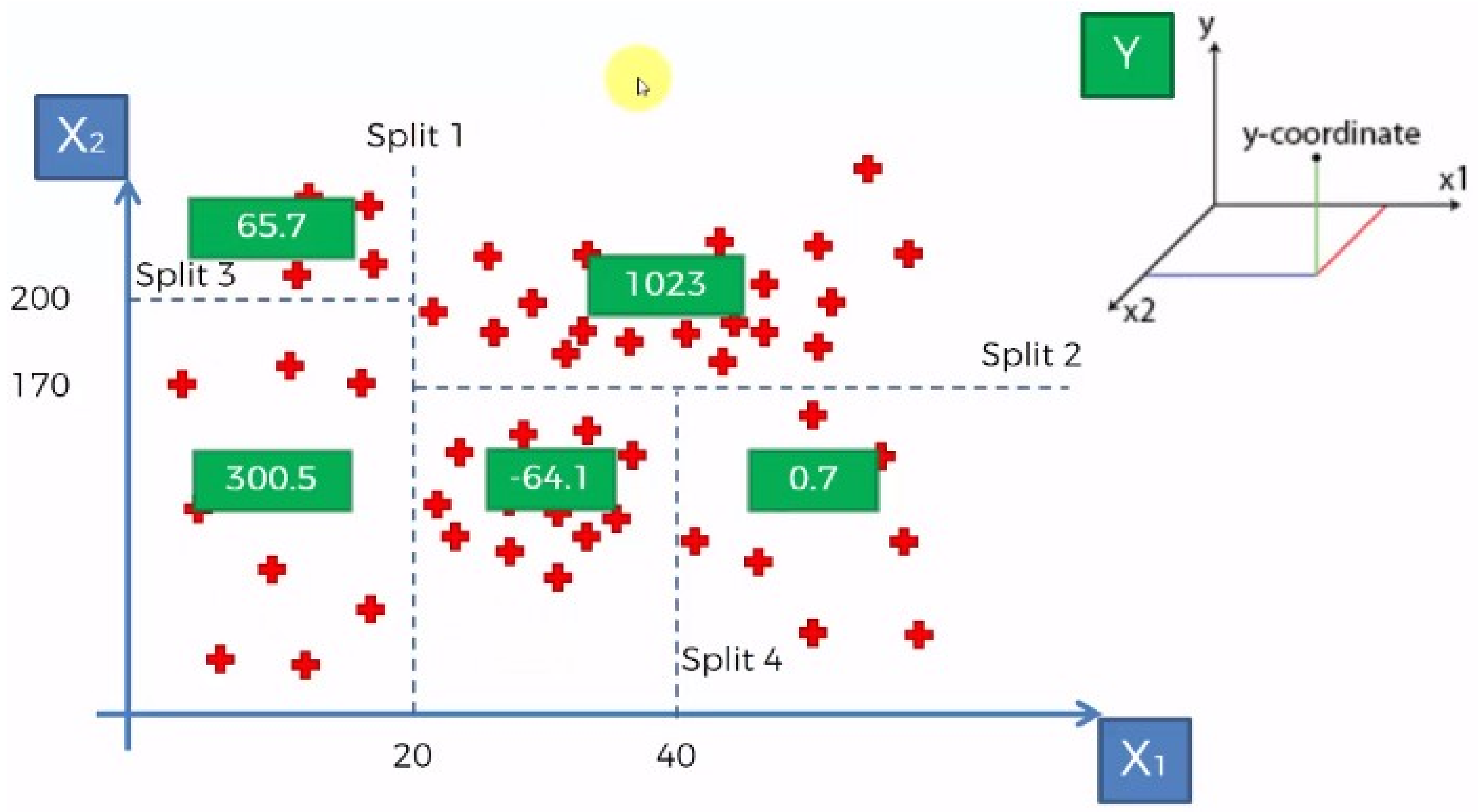
Reference: Super Data Science

Classification Trees



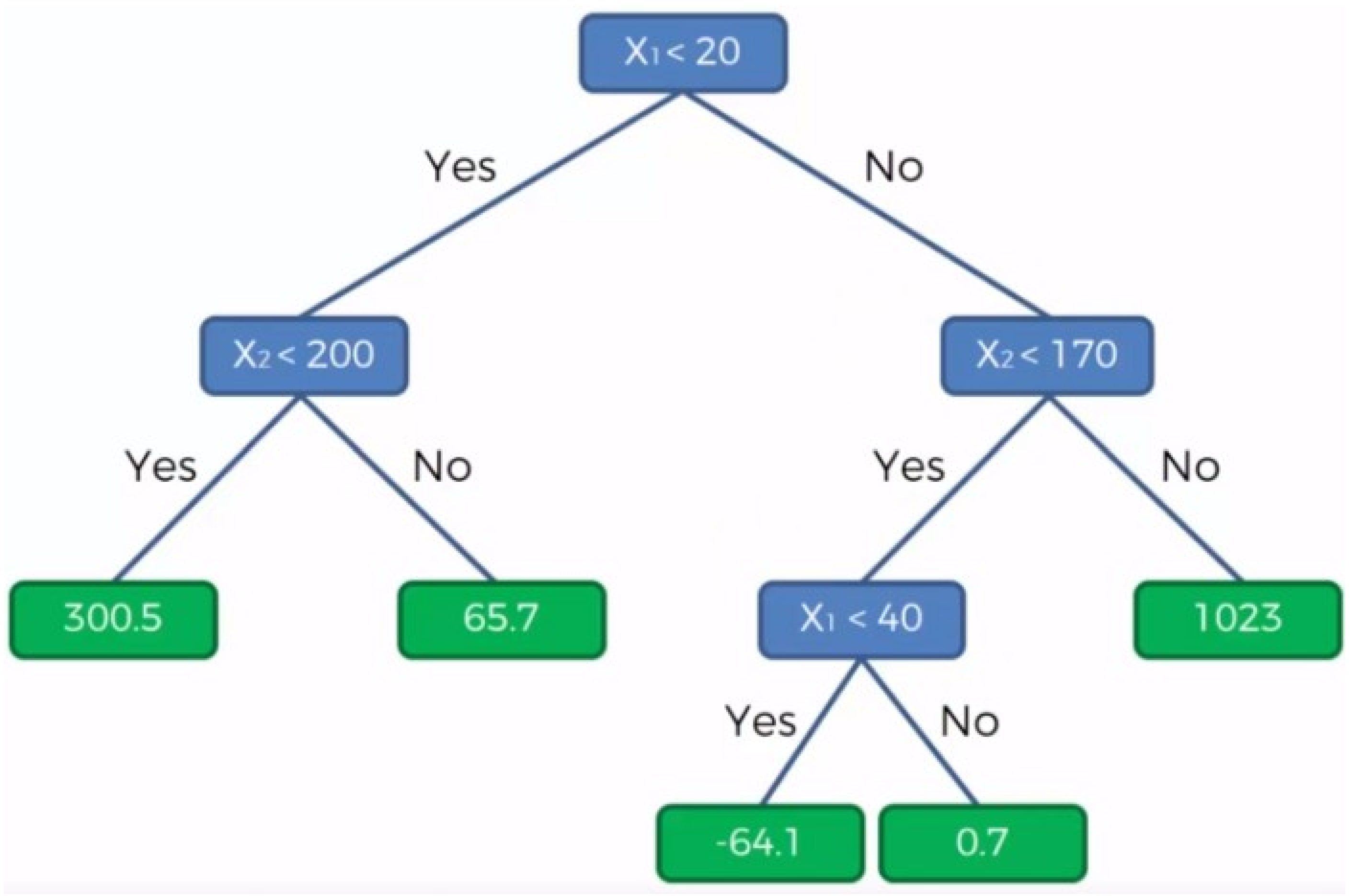
Reference: Super Data Science

Regression Trees



Reference: Super Data Science

Regression Trees



Reference: Super Data Science

Example:

- Practical

Resources

- https://stackabuse.com/
- http://people.sc.fsu.edu
- https://www.geeksforgeeks.org
- http://scikit-learn.org/
- https://machinelearningmastery.com

Thank you

@mitu_skillologies

/mITuSkillologies

@mitu_group

/company/mitu-skillologies

MITUSkillologies

Web Resources
https://mitu.co.in
http://tusharkute.com

contact@mitu.co.in

tushar@tusharkute.com