DECISION TREES

Prepared By,
SHELLY SHIJU GEORGE
ASSISTANT PROFESSOR

Understanding decision trees

- Decision tree learners are powerful classifiers, which utilize a tree structure to model the relationships among the features and the potential outcomes.
- As illustrated in the following figure, this structure earned its name due to the fact that it mirrors how a literal tree begins at a wide trunk, which if followed upward, splits into narrower and narrower branches.
- In much the same way, a decision tree classifier uses a structure of branching decisions, which channel examples into a final predicted class value.

Understanding decision trees (Continued)

– To better understand how this works in practice, let's consider the following tree, which predicts whether a job offer should be accepted.

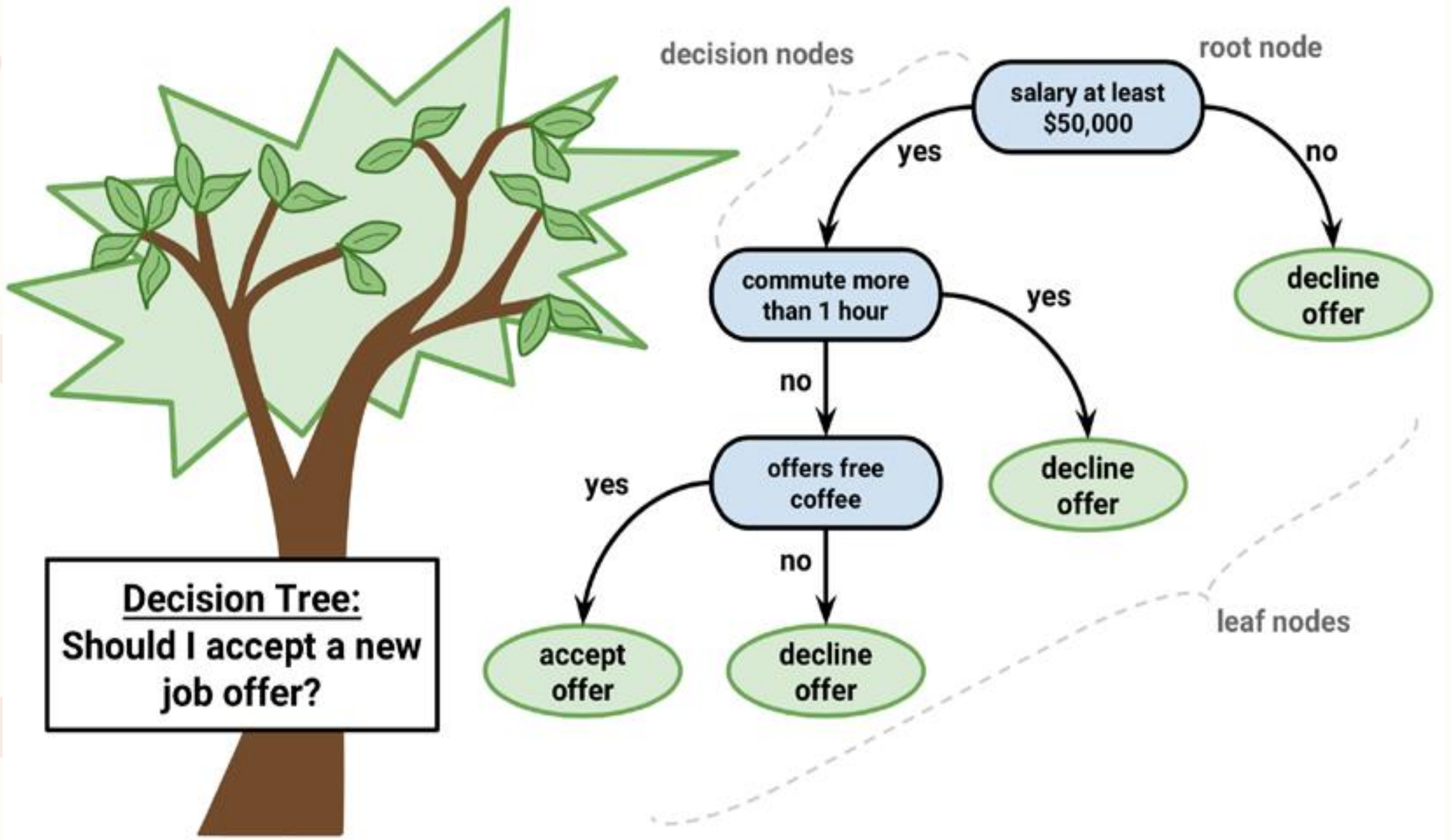
– A job offer to be considered begins at the **root node**, where it is then passed through **decision nodes** that require choices to be made based on the attributes of the job.

– These choices split the data across **branches** that indicate potential outcomes of a decision, depicted here as yes or no outcomes, though in some cases there may be more than two possibilities.

– In the case a final decision can be made, the tree is terminated by **leaf nodes** (also known as **terminal nodes**) that denote the action to be taken as the result of the series of decisions.

– In the case of a predictive model, the leaf nodes provide the expected result given the series of events in the tree.

Understanding decision trees (Continued)



Understanding decision trees (Continued)

- A great benefit of decision tree algorithms is that the flowchart-like tree structure is not necessarily exclusively for the learner's internal use.
- After the model is created, many decision tree algorithms output the resulting structure in a human-readable format.

Understanding decision trees (Continued)

- This provides tremendous insight into how and why the model works or doesn't work well for a particular task.
- This also makes decision trees particularly appropriate for applications in which the classification mechanism needs to be transparent for legal reasons, or in case the results need to be shared with others in order to inform future business practices.

Understanding decision trees (Continued)

With this in mind, some potential uses include:

- Credit scoring models in which the criteria that causes an applicant to be rejected need to be clearly documented and free from bias
- Marketing studies of customer behavior such as satisfaction or churn, which will be shared with management or advertising agencies
- Diagnosis of medical conditions based on laboratory measurements, symptoms, or the rate of disease progression

Divide and conquer

- Decision trees are built using a heuristic called **recursive partitioning**.
- This approach is also commonly known as **divide and conquer** because it splits the data into subsets, which are then split repeatedly into even smaller subsets, and so on and so forth until the process stops when the algorithm determines the data within the subsets are sufficiently homogenous, or another stopping criterion has been met.

Divide and conquer (Continued)

- At first, the root node represents the entire dataset, since no splitting has transpired.
- Next, the decision tree algorithm must choose a feature to split upon; ideally, it chooses the feature most predictive of the target class.
- The examples are then partitioned into groups according to the distinct values of this feature, and the first set of tree branches are formed.

Divide and conquer (Continued)

– Working down each branch, the algorithm continues to divide and conquer the data, choosing the best candidate feature each time to create another decision node, until a stopping criterion is reached.

Divide and conquer might stop at a node in a case that:

– All (or nearly all) of the examples at the node have the same class

– There are no remaining features to distinguish among the examples

– The tree has grown to a predefined size limit

Divide and conquer (Continued)

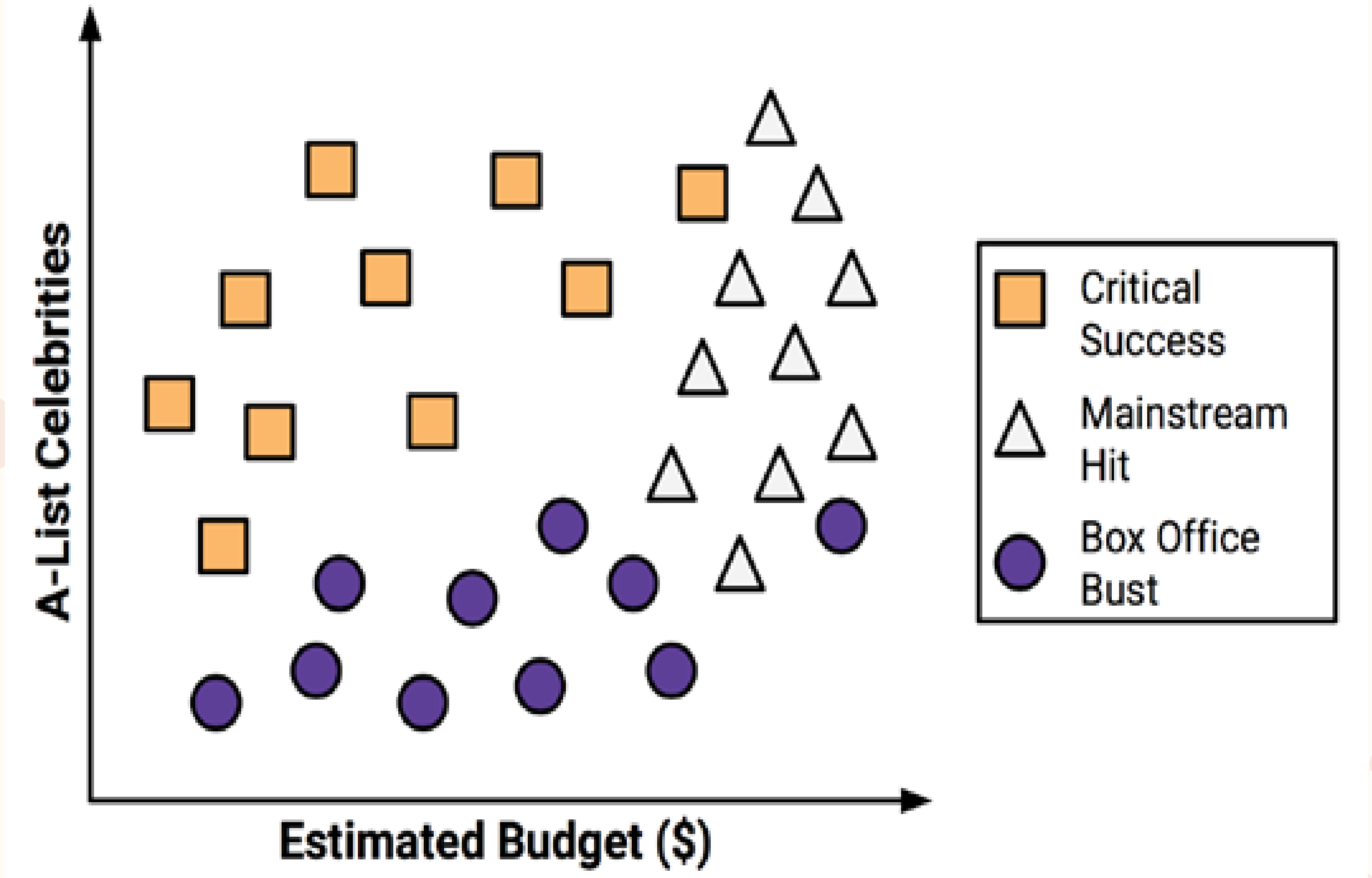
To develop a decision tree algorithm to predict whether a potential movie would fall into one of three categories:

- Critical Success,
- Mainstream Hit, or
- Box Office Bust.

Divide and conquer (Continued)

- To build the decision tree, you turn to the studio archives to examine the factors leading to the success and failure of the company's 30 most recent releases.
- You quickly notice a relationship between the film's estimated shooting budget, the number of A-list celebrities lined up for starring roles, and the level of success.
- Excited about this finding, you produce a scatterplot to illustrate the pattern:

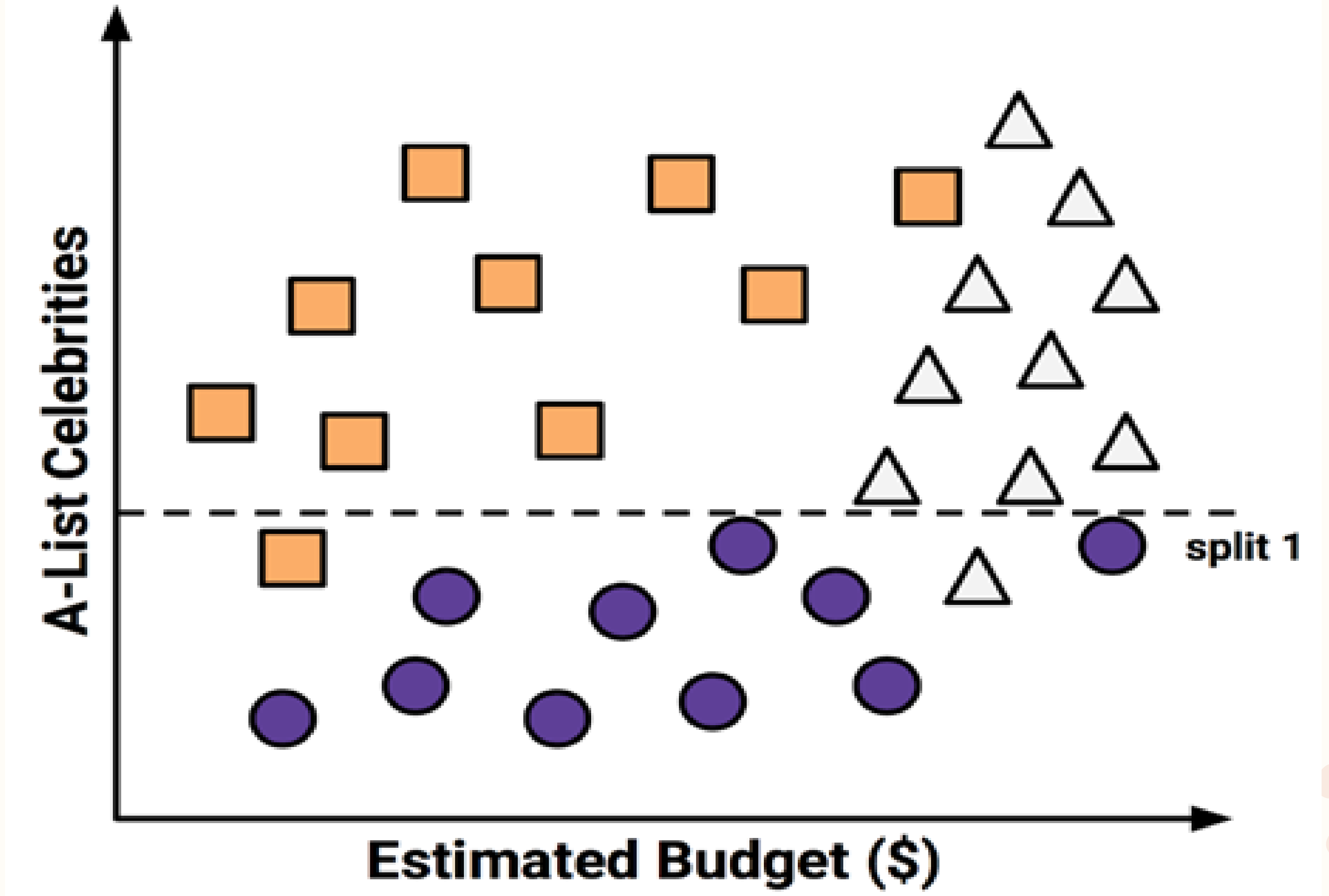
Divide and conquer (Continued)



Divide and conquer (Continued)

- Using the divide and conquer strategy, we can build a simple decision tree from this data.
- First, to create the tree's root node, we split the feature indicating the number of celebrities, partitioning the movies into groups with and without a significant number of A-list stars:

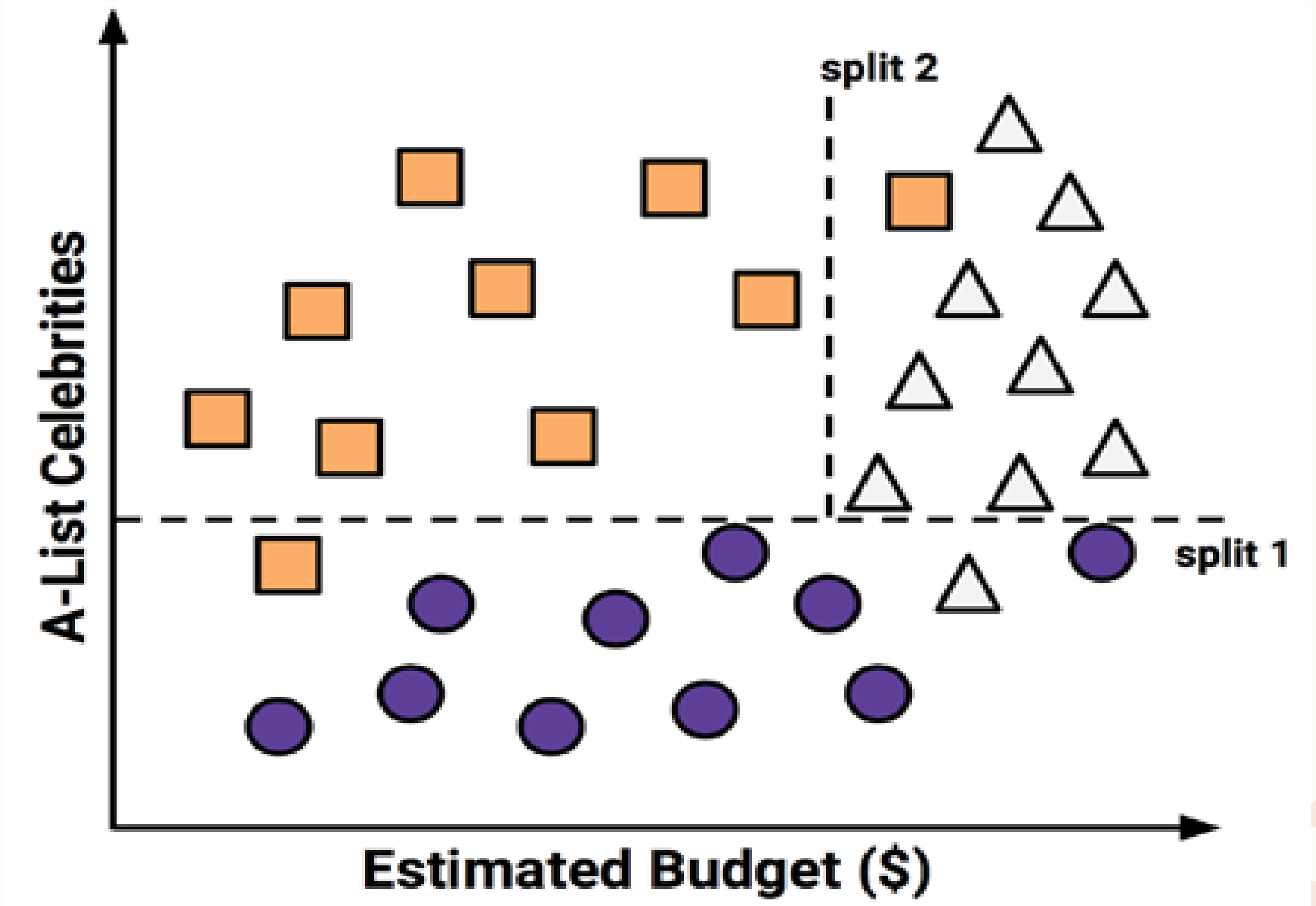
Divide and conquer (Continued)



Divide and conquer (Continued)

- Next, among the group of movies with a larger number of celebrities, we can make another split between movies with and without a high budget:

Divide and conquer (Continued)



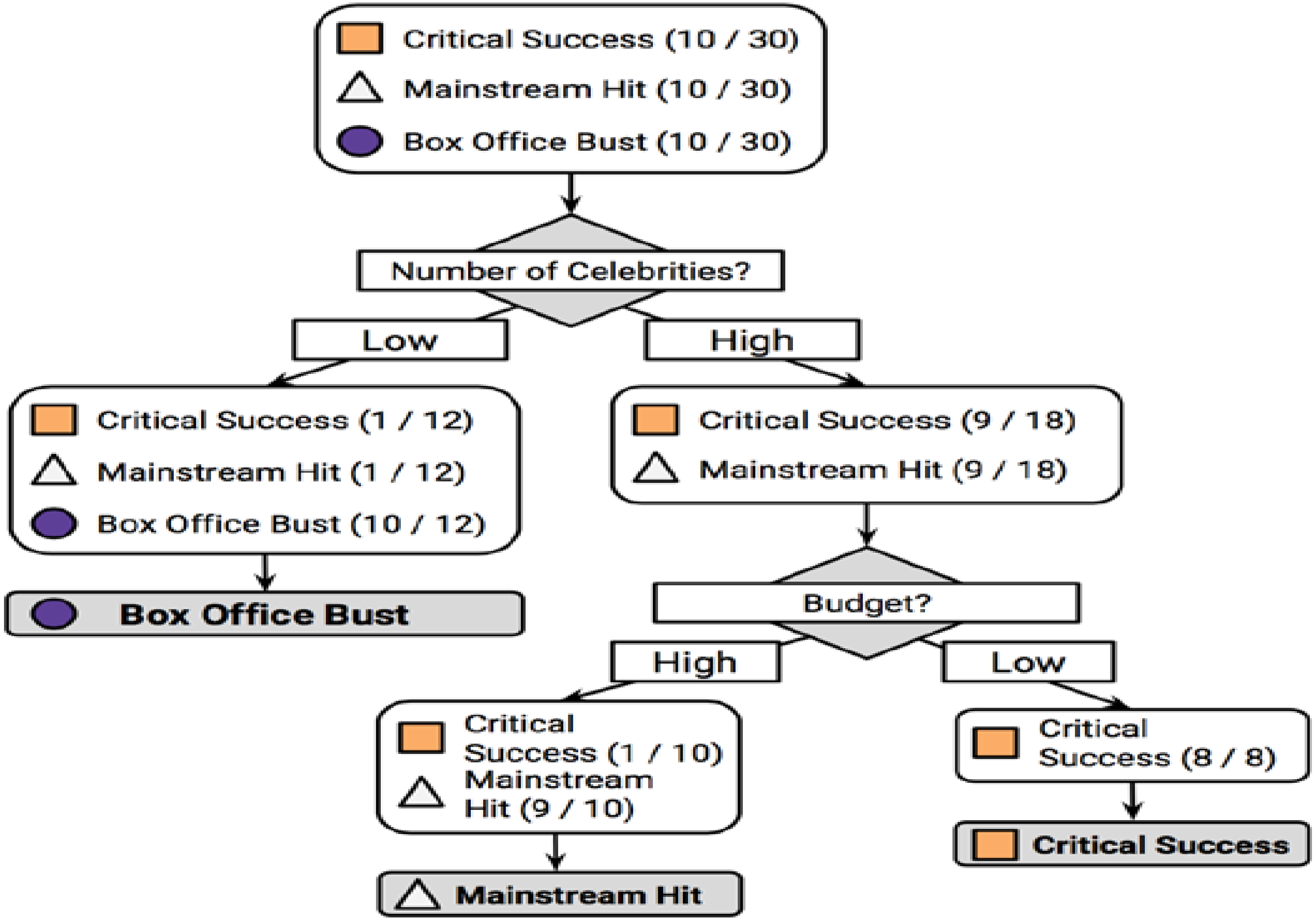
Divide and conquer (Continued)

- At this point, we have partitioned the data into three groups.
- The group at the top-left corner of the diagram is composed entirely of critically acclaimed films.
- This group is distinguished by a high number of celebrities and a relatively low budget.
- At the top-right corner, majority of movies are box office hits with high budgets and a large number of celebrities.
- The final group, which has little star power but budgets ranging from small to large, contains the flops.

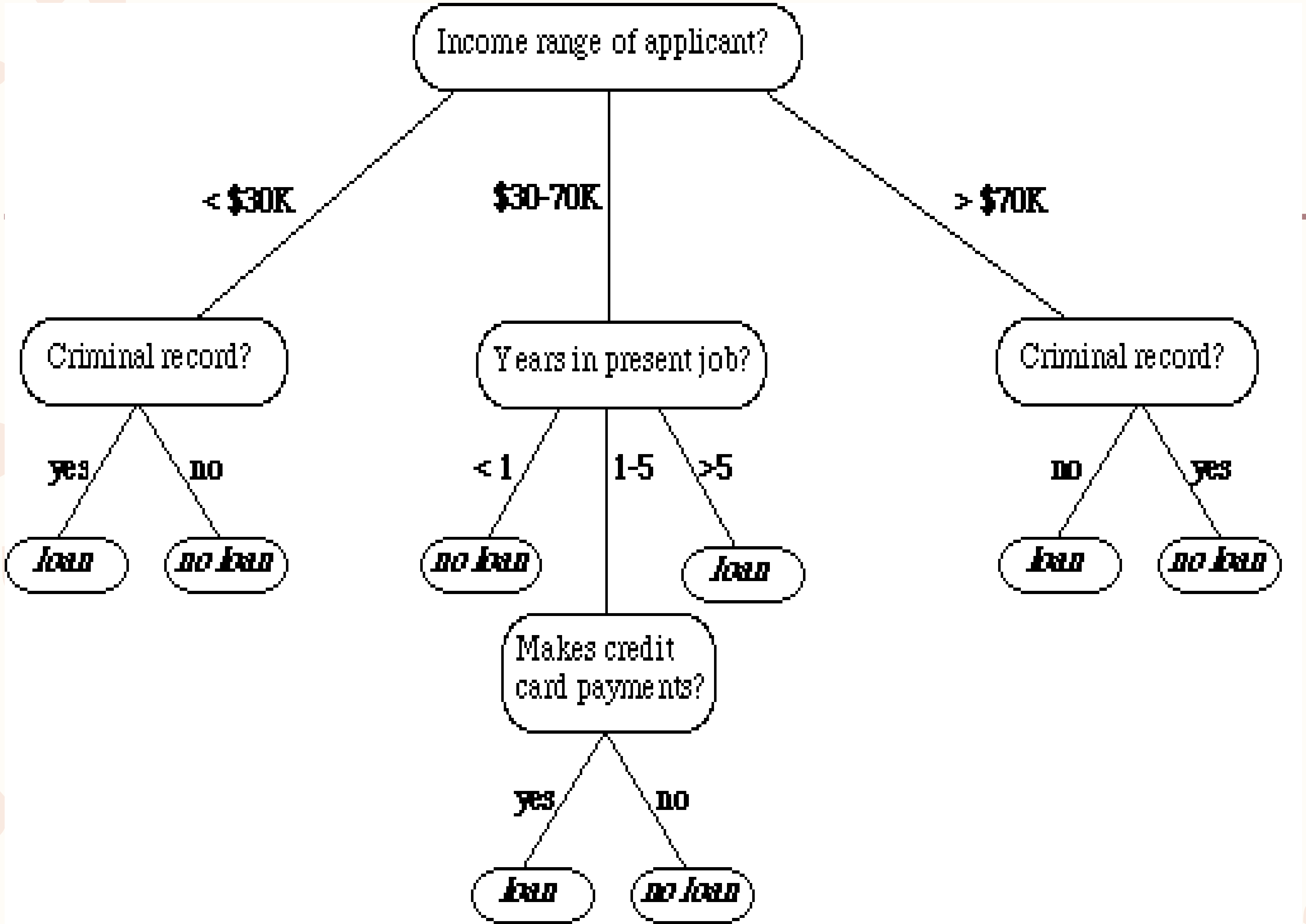
Divide and conquer (Continued)

- Our model for predicting the future success of movies can be represented in a simple tree, as shown in the following diagram.
- To evaluate a script, follow the branches through each decision until the script's success or failure has been predicted.
- In no time, you will be able to identify the most promising options among the backlog of scripts and get back to more important work, such as writing an Academy Awards acceptance speech.

Divide and conquer (Continued)



Divide and conquer (Continued)



The C5.0 decision tree algorithm

- There are numerous implementations of decision trees, but one of the most well-known implementations 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 Iterative Dichotomiser 3 (ID3) algorithm.

The C5.0 decision tree algorithm (Continue)

- The C5.0 algorithm has become the industry standard to produce decision trees, because it does well for most types of problems directly out of the box.
- Compared to other advanced machine learning models, Black Box Methods – Neural Networks and Support Vector Machines, the decision trees built by C5.0 generally perform nearly as well, but are much easier to understand and deploy.

The C5.0 decision tree algorithm (Continue)

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

The C5.0 decision tree algorithm (Continue)

Choosing the best split

- The first challenge that a decision tree will face is to identify which feature to split upon.
- In the previous example, we looked for a way to split the data such that the resulting partitions contained examples primarily of a single class.
- The degree to which a subset of examples contains only a single class is known as **purity**, and any subset composed of only a single class is called **pure**.

The C5.0 decision tree algorithm (Continue)

- There are various measurements of purity that can be used to identify the best decision tree splitting candidate.
- C5.0 uses **entropy**, a concept borrowed from information theory that quantifies the randomness, or disorder, within a set of class values.
- Sets with high entropy are very diverse and provide little information about other items that may also belong in the set, as there is no apparent commonality.
- The decision tree hopes to find splits that reduce entropy, ultimately increasing homogeneity within the groups.

The C5.0 decision tree algorithm (Continue)

- Typically, entropy is measured in bits.
- If there are only two possible classes, entropy values can range from 0 to 1.
- For n classes, entropy ranges from 0 to $\log_2(n)$.
- In each case, the minimum value indicates that the sample is completely homogenous, while the maximum value indicates that the data are as diverse as possible, and no group has even a small plurality.

The C5.0 decision tree algorithm (Continue)

- In the mathematical notion, entropy is specified as follows:
- In this formula, for a given segment of data (S), the term c refers to the number of class levels and p_i refers to the proportion of values falling into class level i.

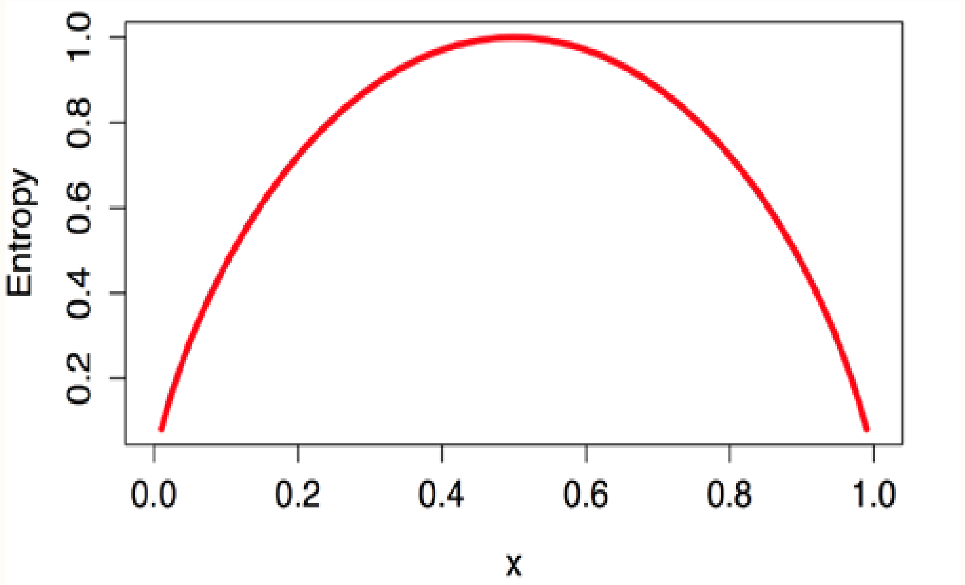
$$\text{Entropy}(S) = \sum_{i=1}^{c} -p_i \, log_2(p_i)$$

The C5.0 decision tree algorithm (Continue)

- For example, suppose we have a partition of data with two classes: red (60 percent) and white (40 percent).
- We can calculate the entropy as follows:
- > -0.60 * $\log_2(0.60)$ - 0.40 * $\log_2(0.40)$
- [1] 0.9709506
- We can examine the entropy for all the possible two-class arrangements.
- If we know that the proportion of examples in one class is x, then the proportion in the other class is (1 – x).

The C5.0 decision tree algorithm (Continue)

- Using the curve() function, we can then plot the entropy for all the possible values of x:
- This results in the following figure:
- As illustrated by the peak in entropy at x = 0.50, a 50-50 split results in maximum entropy.
- As one class increasingly dominates the other, the entropy reduces to zero.



The C5.0 decision tree algorithm (Continue)

– To use entropy to determine the optimal feature to split upon, the algorithm calculates the change in homogeneity that would result from a split on each possible feature, which is a measure known as **information gain**.

– The information gain for a feature F is calculated as the difference between the entropy in the segment before the split (S_1) and the partitions resulting from the split (S_2):

$$\text{InfoGain}(F) = \text{Entropy}(S_1) - \text{Entropy}(S_2)$$

The C5.0 decision tree algorithm (Continue)

- One complication is that after a split, the data is divided into more than one partition.
- Therefore, the function to calculate Entropy(S_2) needs to consider the total entropy across all of the partitions.
- It does this by weighing each partition's entropy by the proportion of records falling into the partition.
- This can be stated in a formula as:

$$\text{Entropy}(S) = \sum_{i=1}^{n} w_i \, \text{Entropy}(P_i)$$

The C5.0 decision tree algorithm (Continue)

- In simple terms, the total entropy resulting from a split is the sum of the entropy of each of the n partitions weighted by the proportion of examples falling in the partition (w_i).

The C5.0 decision tree algorithm (Continue)

- The higher the information gain, the better a feature is at creating homogeneous groups after a split on this feature.
- If the information gain is zero, there is no reduction in entropy for splitting on this feature.
- On the other hand, the maximum information gain is equal to the entropy prior to the split.
- This would imply that the entropy after the split is zero, which means that the split results in completely homogeneous groups.

The C5.0 decision tree algorithm (Continue)

- The previous formulae assume nominal features, but decision trees use information gain for splitting on numeric features as well.
- To do so, a common practice is to test various splits that divide the values into groups greater than or less than a numeric threshold.
- This reduces the numeric feature into a two-level categorical feature that allows information gain to be calculated as usual.
- The numeric cut point yielding the largest information gain is chosen for the split.

The C5.0 decision tree algorithm (Continue)

Pruning the decision tree

- A decision tree can continue to grow indefinitely, choosing splitting features and dividing the data into smaller and smaller partitions until each example is perfectly classified or the algorithm runs out of features to split on.
- However, if the tree grows overly large, many of the decisions it makes will be overly specific and the model will be overfitted to the training data.
- The process of **pruning** a decision tree involves reducing its size such that it generalizes better to unseen data.

The C5.0 decision tree algorithm (Continue)

- One solution to this problem is to stop the tree from growing once it reaches a certain number of decisions or when the decision nodes contain only a small number of examples.
- This is called **early stopping or pre-pruning** the decision tree.
- As the tree avoids doing needless work, this is an appealing strategy.
- However, one downside to this approach is that there is no way to know whether the tree will miss subtle, but important patterns that it would have learned had it grown to a larger size.

The C5.0 decision tree algorithm (Continue)

- An alternative, called **post-pruning**, involves growing a tree that is intentionally too large and pruning leaf nodes to reduce the size of the tree to a more appropriate level.
- This is often a more effective approach than pre-pruning, because it is quite difficult to determine the optimal depth of a decision tree without growing it first.
- Pruning the tree later on allows the algorithm to be certain that all the important data structures were discovered.

The C5.0 decision tree algorithm (Continue)

- One of the benefits of the C5.0 algorithm is that it is opinionated about pruning— it takes care of many decisions automatically using fairly reasonable defaults.
- Its overall strategy is to post-prune the tree. It first grows a large tree that overfits the training data.
- Later, the nodes and branches that have little effect on the classification errors are removed. In some cases, entire branches are moved further up the tree or replaced by simpler decisions.
- These processes of grafting branches are known as **subtree raising and subtree replacement**, respectively.