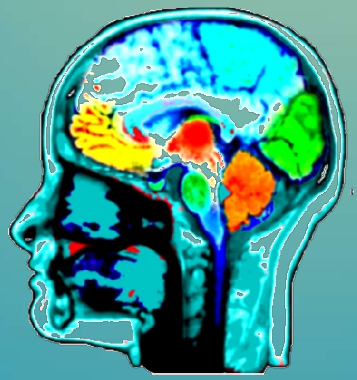
Introduction To Data Mining

Isfahan University of Technology (IUT)

Bahman 1401



Classification

Dr. Hamidreza Hakim
hamid.hakim.u@gmail.com


بسم الله الرحمن الرحيم

Content

میخاهیم از داده ها یه چیزی یاد بگیریم
chapter8! of book

Classification: Basic Concepts

یکی از روش های معروف دسته بندی داده ها به نام درخت تصمیم

Decision Tree Induction

در مسائل ماشین لرنینگ
دو دسته مساله داریم

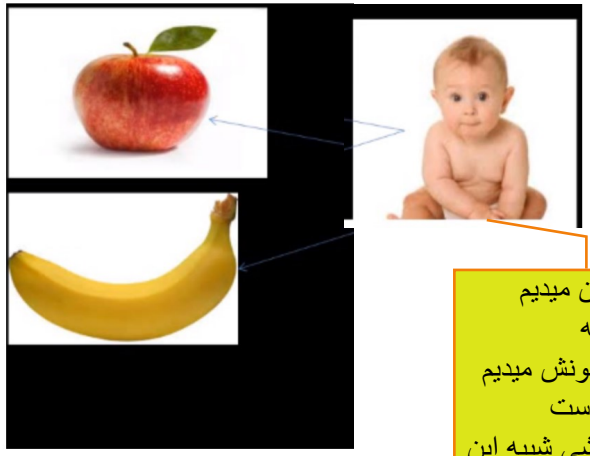
Supervised vs. Unsupervised Learning

به ماشین یه سری داده میگیم و بهش میگیم که هرکدوم از این ابجکت ها چین ینی ابجکت هارا معرفی میکنیم بش ینی لیبل و تارگت را بهش میگیم

- Supervised learning (classification)
- Unsupervised learning (clustering)

در کلسیفیکیشن: شی را با یه لیبل به مدل نشون بدیم تا یاد بگیره بعد یه سری داده ی تست میدیم که لیبل نداره
اگه توانست تشخیص بده که چه لیبلی براش درست است مدل خوبی داریم

دیدی نداریم نسبت به داده ها ینی یه سری ابجکت داریم به مدل میگیم یه طوری برامون دسته بندیشون کن



به بچه نشون میدیم هرچیزی چیه سیب رو نشونش میدیم میگیم سیب است پس اگه یه شی شبیه این نشونش بدیم میفهمه که سیب است

https://gowthamy.medium.com/mach d-learning-vs-unsupervised-learning-f1658e12a780

https://www.mehrnews.com 04

کلاسترینگ :مثل اینکه وارد تالار جشن بشیم و بگیم ادم هایی که توی سالن نشستند را دسته بندی کن
مثلا میتونه بگه ادم های عروس و فامیل های داماد و دوست های عروس و...
در این دسته دیگه ابجکت هارا به مدل معرفی نمیکنیم و به خودش میسپاریم تا بفهمه
از خودش میخایم تا دسته بندی کنه داده ها را

Supervised vs. Unsupervised Learning

- Supervised learning (classification)
  - Supervision: The training data (observations, measurements, etc.) are accompanied by **labels** indicating the class of the observations
  - New data is classified based on the training set
- Unsupervised learning (clustering)
  - The class labels of training data is unknown
  - Given a set of measurements, observations, etc. with the aim of establishing the existence of classes or clusters in the data

یه سری اندازه گیری و مشاهدات داریم و براساس اونها میخایم تصمیم گیری کنیم

Problems: Classification vs. Prediction

- Classification
  - predicts categorical class labels (discrete or nominal)
  - classifies data (constructs a model) based on the training set and the values (class labels) in a classifying attribute and uses it in classifying new data
- Prediction
  - models continuous-valued functions, i.e., predicts unknown or missing values

در supervised learning دوتا مساله اصلی وجود داره:
بعضی وقت ها برچسبی که میخایم بزنیم به داده ها مشخص و discrete است مثل وقتی که میگیم هوا دوحالت داره : سرد است یا گرم
یه جاهایی از مدل میخایم یه طیف را برامون پیشبینی کنه مثلا میخایم دما را برامون پیشبینی کنه
ما اطلاعات دمای روزهای قبل را داریم مثل گرما و میزان رطوبت و تابش خورشید و مکان جغرافیایی و ... و میگیم دما توی این نقطه باتوجه به این اطلاعاتی که بت دادیم اینقدره حالا بیا دما را توی نقطه ی جدید که ازت میپرسیم پیشبینی کن
ینی میخایم چیزی را پیشبینی کنی که پیوسته است یه تعداد مشخصی از حالت را نداره اگه تعداد حالت داشته باشه میگیم میخایم دسته بندی کنیم که میشه classification
اگه تعداد دسته نداشته باشیم و یه طیف داشته باشیم میشه پیشبینی یا prediction

Prediction Problems: Classification vs. Prediction

- Typical applications
  - Credit/loan approval:
  - Medical diagnosis: if a tumor is cancerous or benign
  - Fraud detection: if a transaction is fraudulent
  - Web page categorization: which category it is

به یکی وام بدیم یانه؟
دوحالته پس تعداد گسسته
و مشخص دسته داریم
پس classification

فردی سرطان داره یانه؟

Fraud detection is typically considered a classification task, as the goal is to determine whether or not a given transaction or activity is fraudulent. In a classification task, the model is trained to assign input data to one or more categories (in this case, "fraudulent" or " not fraudulent") based on patterns and relationships in the training data.

Prediction, on the other hand, generally refers to a task where the aim is to predict a numerical value, such as a stock price or a customer's lifetime value. While fraud detection might involve making predictions about the likelihood of fraud occurring, the ultimate output is still a binary classification: fraudulent or not fraudulent.

Web page categorization can also be considered a classification task, as the goal is to assign web pages to one or more categories based on their content and characteristics. In this case, the categories might include topics such as "sports," " entertainment," "business," etc.

The process of web page categorization typically involves training a machine learning model using labeled data - that is, a set of web pages that have already been manually assigned to their respective categories. The model then uses patterns and relationships in the training data to predict the category of new, unlabeled web pages.

Like other classification tasks, web page categorization can be approached using various machine learning algorithms, including decision trees, support vector machines, and neural networks.

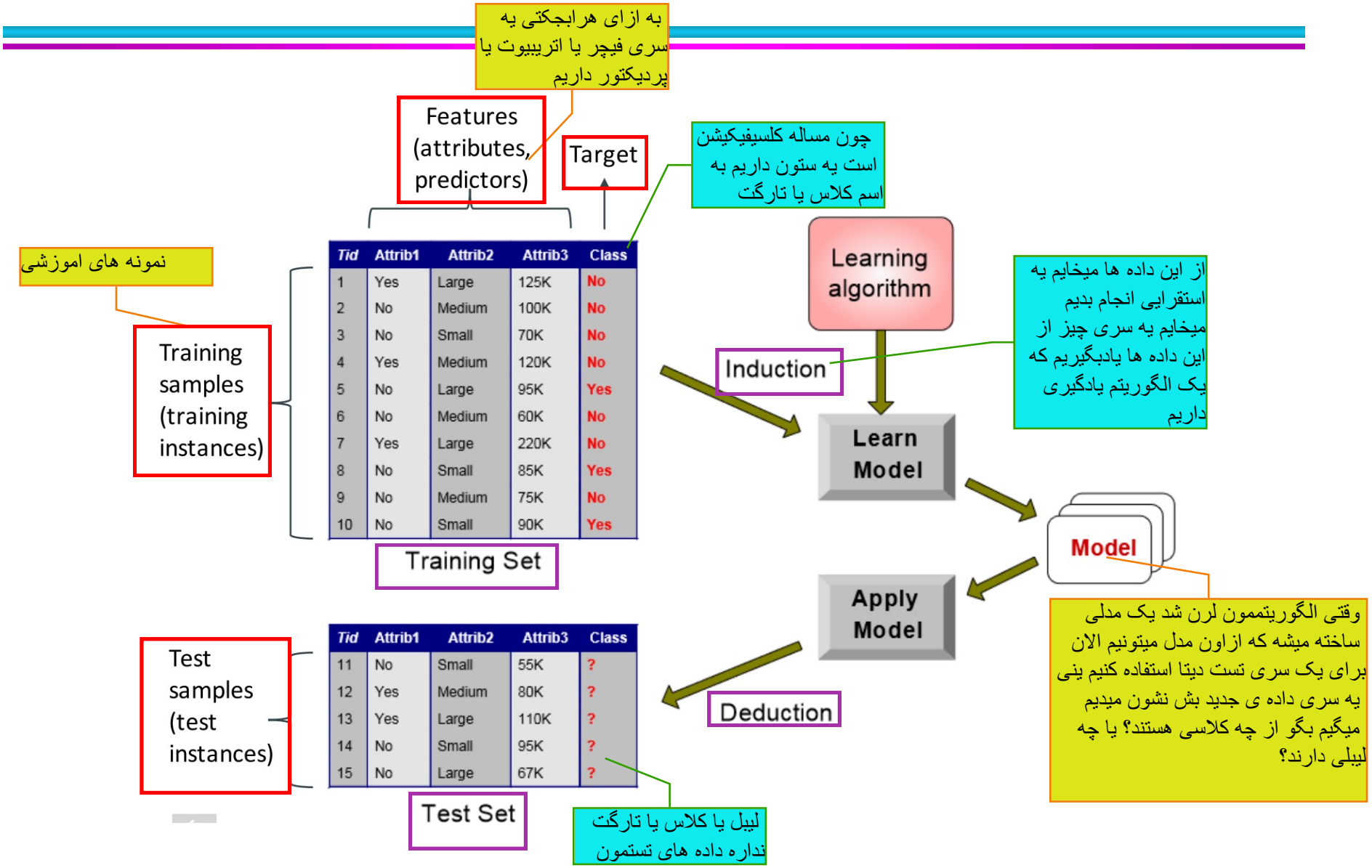
Examples of Classification Task

Task	Attribute set, x	Class label, y
Categorizing email messages	Features extracted from email message header and content	spam or non-spam
Identifying tumor cells	Features extracted from x-rays or MRI scans	malignant or benign cells
Cataloging galaxies	Features extracted from telescope images	Elliptical, spiral, or irregular-shaped galaxies

فهرست نویسی کهکشان ها

کهکشان های بیضوی، مارپیچی یا نامنظم

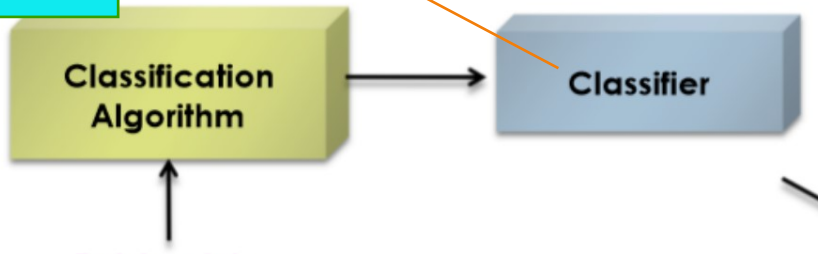
Classification Process



Training phase , Model construction

دوتا فاز اصلی برای کلسیفیکیشن داریم
فاز اول: training phase
یا فاز ساخت مدل ینی داده هارا دست یک کلسیفیکشن الگوریتم میدیم داده هایی که لیبل و هدفشون مشخص است نیازی به ستون ایدی یا ایندکس نیست اینجا
این کلسیفیکیشن الگوریتم میاد بین داده ها یه فرمولی درمیاره

میتونه مثل قانون باشه
مثلا میخاد بفهمه یه نفر با فلان شرایط ایا وامش را پرداخت میکنه یا نه؟
براساس وضعیت سن و درامدش میگیم پرداخت میکنه یا نه
پیشبینی فاز دوم در کلسیفیکیشن است



If age <30 & income <30k
Then, Budget-Spender

If age <30 & income > 30k
Then, Big-Spender

If 30<age<60 and income >30k
Then Big-spender

If 30<age<60 and income <30k
Then Budget-spender

If age >60, Then Budget-spender

Training data

Age	Income	Class label
27	28K	Budget-Spender
35	36K	Big-Spender
38	28K	Budget-Spender
65	45K	Budget-Spender
20	18k	Budget-Spender
75	40k	Budget-Spender
28	50k	Big-Spender
40	60k	Big-Spender
60	65k	Big-Spender

random_state is a parameter in Scikit-learn's train_test_split function that allows you to specify the random seed used by the random number generator.

When you split your data into training, validation, and test sets, it's important to ensure that the same random samples are selected each time you run your code. This can be achieved by setting a fixed value for random_state.

In the example I provided earlier, we set random_state=42. This means that if you run the code multiple times with the same dataset and the same random_state value, you will get the same splits each time. This is useful for reproducibility and debugging purposes.

Note that the specific value of random_state doesn't matter as long as it's consistent across runs. You can choose any integer value for random_state to achieve this consistency.

Sure! Here's an example of how you can split a dataset into training, validation, and test sets in Python:

```
import pandas as pd
from sklearn.model_selection import train_test_split

# Load the dataset into a Pandas dataframe
df = pd.read_csv('customer_transactions.csv')

# Split the data into training and test sets (70/30 split)
X_train, X_test, y_train, y_test = train_test_split(df.drop('returned', axis=1), df['returned'], test_size=0.3, random_state=42)

# Split the training set further into training and validation sets (50/50 split)
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.5, random_state=42)

# Print the shapes of each set
print(f'Training set shape: {X_train.shape}')
print(f'Validation set shape: {X_val.shape}')
print(f'Test set shape: {X_test.shape}')
```

In this example, we first load the customer transactions dataset into a Pandas dataframe. We then use the train_test_split function from Scikit-learn to split the data into a 70/30 split of training and test sets.

Next, we split the training set further into a 50/50 split of training and validation sets using the same function. Finally, we print out the shapes of each set to confirm that the data has been split correctly.

Note that in this example, we assume that the dataset has a binary classification target variable 'returned' indicating whether a customer returned an item or not. We drop this column from the feature data using drop('returned', axis=1) when splitting into training/validation/test sets. It's also important to set a random_state parameter to ensure reproducibility of the splits.

I generated a sample dataset to use in the previous example. Here is an example of how the dataset could look like:

Copy Code

```
customer_id,age,gender,location,item,purchase_amount,returned
1,25,Female,New York,Clothing,50.0,No
2,32,Male,Los Angeles,Electronics,100.0,Yes
3,45,Female,Chicago,Books,20.0,No
4,38,Male,Miami,Home goods,75.0,No
5,22,Female,Boston,Clothing,30.0,Yes
6,56,Male,San Francisco,Electronics,200.0,No
7,29,Female,Seattle,Home goods,80.0,Yes
8,41,Male,Dallas,Clothing,55.0,No
9,34,Female,Washington D.C.,Books,15.0,Yes
10,48,Male,Houston,Home goods,90.0,No
```

This dataset contains 10 customer transactions with demographic information (age, gender, and location), items purchased, purchase amount, and whether they returned any items. In this example, we split the data into training, validation, and test sets to build a machine learning model that predicts whether a customer returned an item based on their demographic information and purchase history.

Copy Code

```
Training set shape: (3, 6)
Validation set shape: (4, 6)
Test set shape: (3, 6)
```

The output shows the number of rows and columns in each set after splitting the data into training, validation, and test sets. Note that the actual shapes may differ depending on the size of your original dataset and the random seed used in the split.

In this example, we split the data into a 70/30 split of training and test sets using `test_size=0.3` in the `train_test_split` function. Then, we further split the training set into a 50/50 split of training and validation sets using the same function with `test_size=0.5`.

The output shows that the training set has 3 rows and 6 columns, the validation set has 4 rows and 6 columns, and the test set has 3 rows and 6 columns.

Evaluation and usage phases

داده های تست را اینطوری تعریف کردیم:
داده هایی که مدل قبلا ندیده

فاز کارایی
میخایم داده های تستمون را بش بدیم ببینیم چه نتیجه ای میده؟

یه عملکردی به ما میده ما میخایم یه بازخوردی از مدلمون بگیریم ازش میپرسیم یاد گرفتی یا نه؟ اگه یادنگرفته نحوه ی اموزشمون را مثلا عوض کنیم یا داده ی بیشتری بش بدیم تا بهتر یادبگیره یا مدلمون را عوض کنیم

1-Test the classifier

این تست دیتاها را از قبل بش نشون ندیدم ها!!!!!!!! پنها دیتا های جدیده که مدل ندیدده

Test Data

Age	Income	Class label
27	28K	Budget-Spenders
25	36K	Big-Spenders
70	45K	Budget-Spenders
40	35k	Big-Spender

Accuracy

به این عملیات میگن ارزیابی یا evaluation

فیدبک میگیریم از مدلمون

2-If acceptable accuracy

Unlabeled data

Age	Income
18	28K
37	40K
60	45K
40	36k

Classified data

Age	Income	Class label
18	28K	Budget-Spenders
37	40K	Big-Spenders
60	45K	Budget-Spenders
40	36k	Budget-Spenders

Evaluation and usage phases

مدلمون را چطوری ارزیابی کنیم؟
چندین روش برای ارزیابی کردن هست
اولین روش: پیداکردن میزان دقت یا accuracy
دومین روش: استفاده از ماتریس کانفیوژن confusion matrix
سومین معیار: error rate
تقسیم تعداد اشتباه ها به کل تعداد پیشبینی شده

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}.$$

True positive =TP

Table 3.4. Confusion matrix for a binary classification problem.

لیبل یا تارگت یا کلاس هامون دوحالت دارند یک یا صفر

		Predicted Class	
		Class = 1	Class = 0
Actual Class	Class = 1	f_{11}	f_{10}
	Class = 0	f_{01}	f_{00}

False Negative=FN

ایده ال ما اینه که یک ها را یک تشخیص بده و صفر ها را صفر

False Positive=FP

True negative =TN

$$\text{Error rate} = \frac{\text{Number of wrong predictions}}{\text{Total number of predictions}} = \frac{f_{10} + f_{01}}{f_{11} + f_{10} + f_{01} + f_{00}}.$$

Suppose we have a classification problem where we need to predict whether a customer will buy a product or not based on their demographic information. We have a dataset of 1000 customers, out of which we use 700 for training our model and the remaining 300 for testing. After training our model, we make predictions on the test set and get the following results:

- True positives (TP): 150
- False positives (FP): 30
- True negatives (TN): 100
- False negatives (FN): 20

To calculate the accuracy of our model, we can use the following formula:

Accuracy = (TP + TN) / (TP + TN + FP + FN)

Plugging in the values from our results, we get:

Accuracy = (150 + 100) / (150 + 100 + 30 + 20)
Accuracy = 0.833

This means that our model correctly predicted the outcome for 83.3% of the customers in the test set. We can use this accuracy score to evaluate the performance of our model and compare it with other models or benchmarks.

A confusion matrix is a performance evaluation metric used in machine learning to evaluate the accuracy of a classification model. It is a table that compares the actual values of the target variable with the predicted values of the target variable. The matrix consists of four different combinations of predictions and actual outcomes: true positive (TP), false positive (FP), false negative (FN), and true negative (TN).

Here's an example: let's say we have built a binary classifier that predicts whether an email is spam or not. We have tested the model on a dataset of 100 emails, where 60 are non-spam (class 0) and 40 are spam (class 1). The model has predicted 45 non-spam emails correctly, 15 non-spam emails incorrectly as spam, 30 spam emails correctly, and 10 spam emails incorrectly as non-spam.

The confusion matrix for this example would be:

	Predicted Not Spam	Predicted Spam
Actual Not Spam	TN = 45	FP = 15
Actual Spam	FN = 10	TP = 30

Here, TN stands for true negatives, which represents the number of non-spam emails that were correctly predicted as non-spam. FP stands for false positives, which represents the number of non-spam emails that were incorrectly predicted as spam. FN stands for false negatives, which represents the number of spam emails that were incorrectly predicted as non-spam. Lastly, TP stands for true positives, which represents the number of spam emails that were correctly predicted as spam.

From the confusion matrix, we can calculate several performance metrics such as accuracy, precision, recall, and F1 score. Accuracy measures the overall performance of the model, while precision and recall measure the model's ability to correctly predict positive cases and identify all positive cases, respectively. F1 score is a weighted average of precision and recall. These metrics can help us understand how well our model is performing and identify areas for improvement.

Performance metrics in machine learning are used to evaluate the performance of a machine learning model. These metrics help data scientists to determine how well their model is performing and identify areas for improvement. Different machine learning tasks require different performance metrics. Here are some common performance metrics in machine learning along with examples:

Accuracy: The most basic performance metric that measures the percentage of correctly predicted instances among all the instances in the dataset. It's calculated by dividing the number of correct predictions by the total number of predictions. Example: If a model correctly predicts 90 out of 100 instances, then the accuracy score would be 90%.

Precision: Measures the percentage of true positive predictions among all positive predictions made by the model. It's calculated by dividing the number of true positives by the sum of true positives and false positives. Example: If a model predicts 30 positives and 25 of them are actual positives, then the precision score would be 83.33%.

Recall: Measures the percentage of true positive predictions among all actual positive instances in the dataset. It's calculated by dividing the number of true positives by the sum of true positives and false negatives. Example: If there are 50 actual positives in the dataset and the model predicts 40 of them, then the recall score would be 80%.

F1 Score: The harmonic mean of precision and recall. It provides a balance between precision and recall. It's calculated as 2 * (precision * recall) / (precision + recall). Example: If the precision and recall scores are 0.85 and 0.75 respectively, then the F1 score would be 0.79.

Mean Absolute Error (MAE): A regression performance metric that measures the average absolute difference between the predicted and actual values. Example: If the predicted value is 10 and the actual value is 7, then the absolute difference is 3. If the model makes 10 predictions with absolute differences of 3, then the MAE would be 3/10 = 0.3.

Root Mean Squared Error (RMSE): Another regression performance metric that measures the square root of the average squared difference between the predicted and actual values. Example: If the predicted value is 10 and the actual value is 7, then the squared difference is 9. If the model makes 10 predictions with squared differences of 9, then the RMSE would be the square root of 9/10 = 0.95

Let's say we have a binary classification problem where we want to predict whether a patient has a disease or not. We have a dataset of 100 patients, where 60 patients do not have the disease and 40 patients have the disease. We use a machine learning model to make predictions on this dataset and obtain the following confusion matrix:

	Predicted No	Predicted Yes
Actual No	50	10
Actual Yes	5	35

From this confusion matrix, we can calculate the following metrics:

Accuracy: the proportion of correct predictions out of all predictions made by the model.
Accuracy = (True Positives + True Negatives) / Total Population
Accuracy = (50 + 35) / 100
Accuracy = 0.85
So the accuracy of our model is 85%.

Precision: the proportion of true positives (patients predicted as having the disease who actually have the disease) out of all patients predicted as having the disease.
Precision = True Positives / (True Positives + False Positives)
Precision = 35 / (35 + 10)
Precision = 0.78
So the precision of our model is 78%.

Recall: the proportion of true positives out of all actual positives (patients who actually have the disease).
Recall = True Positives / (True Positives + False Negatives)
Recall = 35 / (35 + 5)
Recall = 0.88
So the recall of our model is 88%.

F1 score: a combined metric that takes into account both precision and recall.
F1 Score = 2 * (Precision * Recall) / (Precision + Recall)
F1 Score = 2 * (0.78 * 0.88) / (0.78 + 0.88)
F1 Score = 0.83
So the F1 score of our model is 0.83.

In machine learning, we typically divide our available data into three sets: training set, validation set, and test set.

The purpose of the training set is to train the model. The purpose of the validation set is to tune hyperparameters or select the best model among multiple possible models. The purpose of the test set is to evaluate the final performance of the model on unseen data.

The key difference between the validation set and the test set is their usage in the machine learning process. The validation set is used during the training process to validate different models' performance on unseen data and select the best performing one. In contrast, the test set is used only once at the end of the training process to evaluate the model's generalization ability on previously unseen data.

Here's an example to illustrate the difference:

Suppose you have a dataset of images of cats and dogs. You can divide this data into training, validation, and test sets. You can use 70% of the data for training (training set), 15% of the data for model selection (validation set), and the remaining 15% for evaluating the final performance of the model (test set).

During the training process, the model is trained on the training set, and its performance is evaluated on the validation set. Multiple models with different hyperparameters can be trained and validated using the validation set, and the best performing model is selected for final evaluation.

Once the final model is selected, it is evaluated on the test set. This provides an unbiased estimate of the model's performance on completely unseen data.

Classification—A Two-Step Process

- Model construction: describing a set of predetermined classes
  - Each tuple/sample is assumed to belong to a predefined class, as determined by the class label attribute
  - The set of tuples used for model construction is training set
  - The model is represented as classification rules, decision trees, or mathematical formulae

شبکه های عصبی

- Model usage: for classifying future or unknown objects
  - Estimate accuracy of the model
    - The known label of test sample is compared with the classified result from the model
    - Accuracy rate is the percentage of test set samples that are correctly classified by the model
    - Test set is independent of training set (otherwise overfitting)
  - If the accuracy is acceptable, use the model to classify new data

قدرت مدل ساخته شده چقدره؟

- Note: If *the test set* is used to select models, it is called validation (test) set

شبکه های عصبی فرمول ریاضی برامون درمیارند در کلسیفیکیشن
مدلمون مثلا یه کلسیفیکیشن رول است ینی ما میخایم قوانین را از داده ها استخراج کنیم

Binary classification vs Multiclass classification

Binary classification

- Classification tasks with only two classes, typically denoted by {+,−}, {+1,−1}, or {Pos, Neg}.
- Example: email spam detection, (pos/neg) sentiment analysis.

Multiclass classification

- Classification tasks with more than two classes.
- Example: email topic detection, (pos/null/neg) sentiment analysis.

سوال یا مساله:
یه نمودار قیمت دلار داریم میخایم ببینیم قیمتش بالا میره یا پایین در روز اینده این که جواب مساله دوتاحالت داره میشه باینری کلسیفیکیشن
میتونیم یه حالت دیگه هم اضافه کنیم به جواب ها قیمت ثابت بماند نسبت به روز قبل
پس الان سه تا جواب داره مساله میشه مولتی کلاس
مسائل کلسیفیکیشن چند حالته را میشه با دوحالته هم حل کرد چطوری؟
مثلا بگیم قیمت ثابت است یا ثابت نیست؟
اول اینو میپرسیم ازش
اگه گفت ثابت که هیچی اگه گفت غیر ثابت میگیم کم یشه یا زیاد؟ اینجا نیاز به یک کلسیفایر دیگه داریم که درباره ی بالا و پایین بودن قیمت تصمیم گیری کنه

Classification Techniques

- Decision Tree based Methods
- Rule-based Methods
- Nearest-neighbor
- Naïve Bayes and Bayesian Belief Networks
- Support Vector Machines
- Neural Networks, Deep Neural Nets
- Ensemble Classifiers
  - Boosting, Bagging, Random Forests
- And many more

ترکیب کلسیفایر های مختلف باهم

DECISION TREE

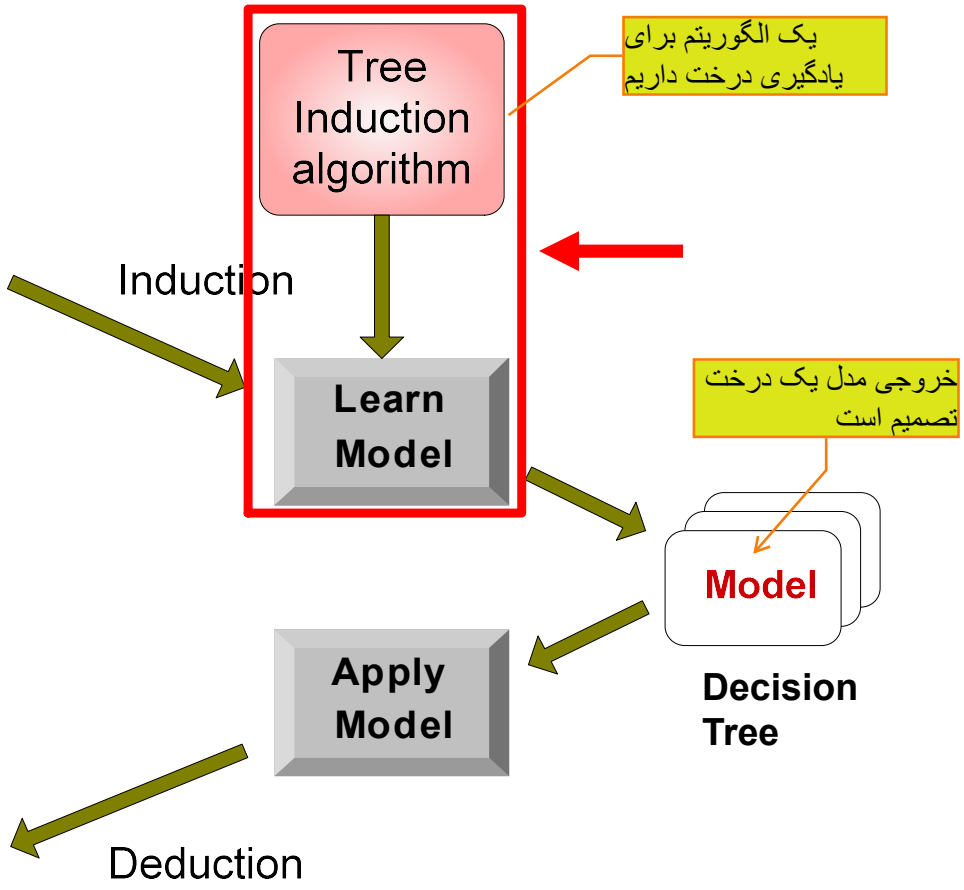
Decision Tree Classification Task

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

Test Set



Example of a Decision Tree

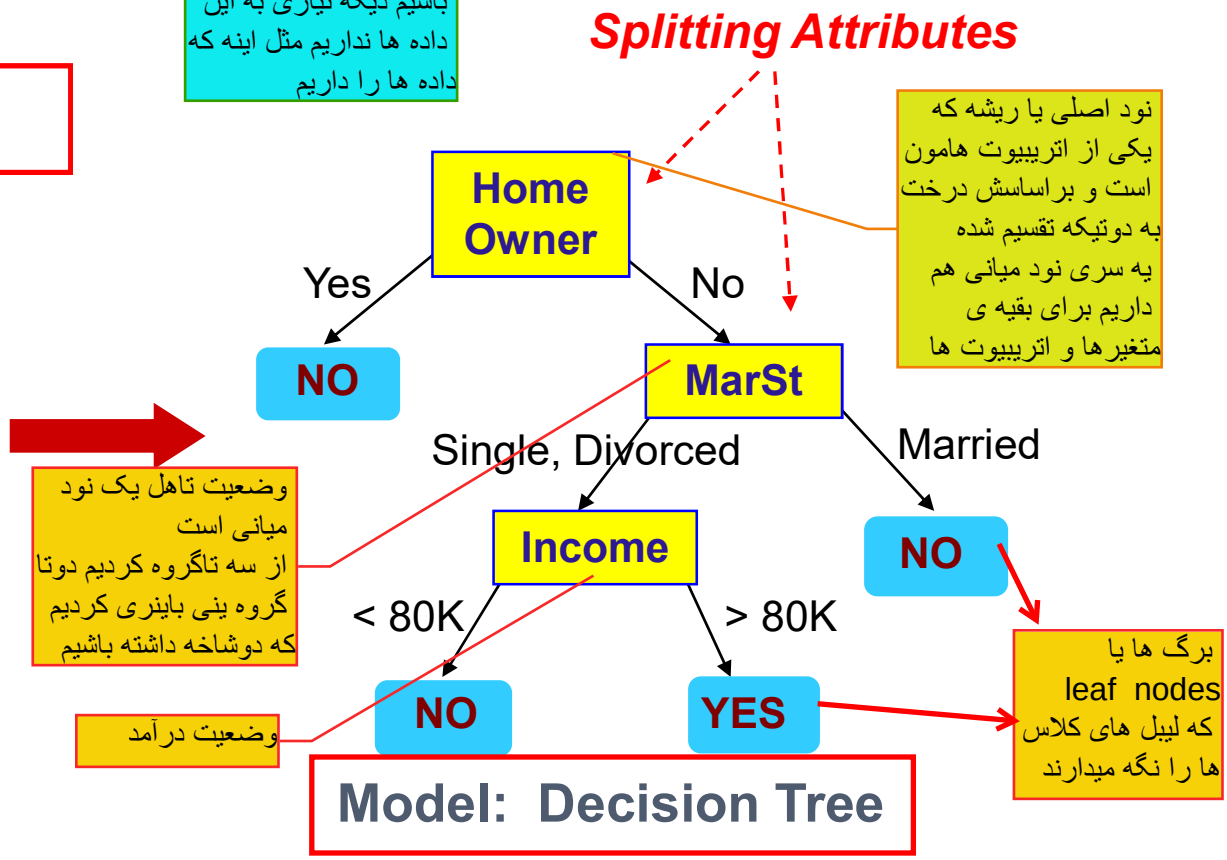
سوال یا مساله:
اگه یه ادمی وام بگیره ایا پس میده یا نه؟

وقتی این درخت را داشته باشیم دیگه نیازی به این داده ها نداریم مثل اینه که داده ها را داریم

categorical | categorical | continuous | class

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

Training Data



نود اصلی یا ریشه که یکی از اتریبیوت هامون است و براساسش درخت به دوتیکه تقسیم شده یه سری نود میانی هم داریم برای بقیه ی متغیرها و اتریبیوت ها

وضعیت تاهل یک نود میانی است از سه تاگروه کردیم دوتا گروه ینی باینری کردیم که دوشاخه داشته باشیم

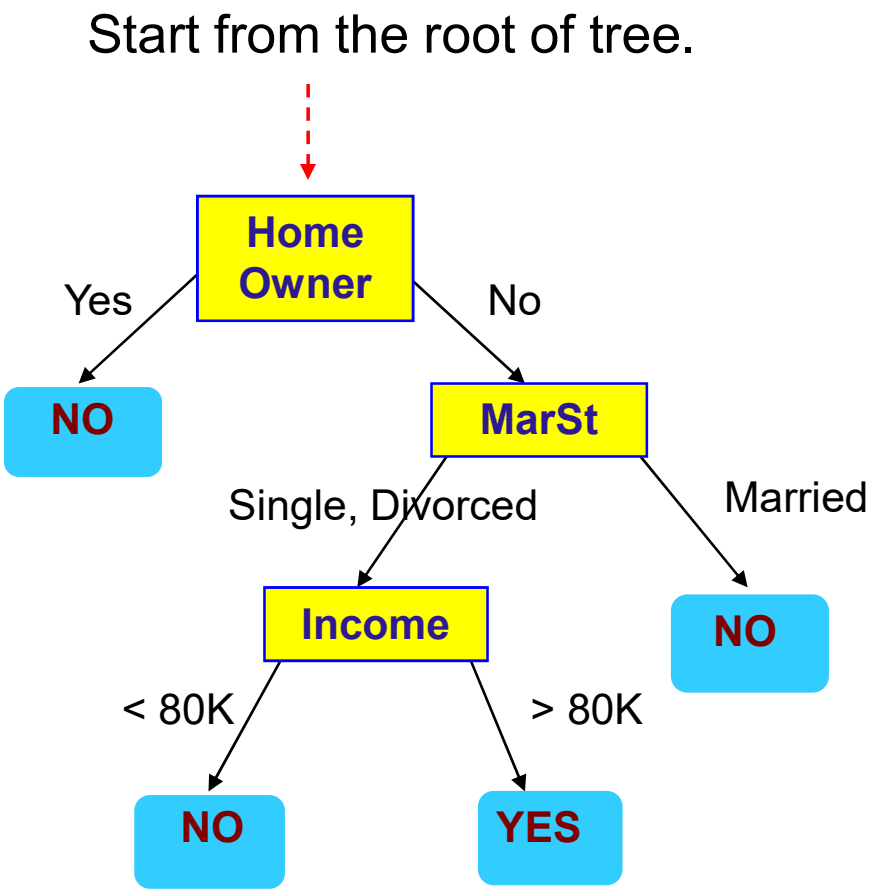
وضعیت درآمد

برگ ها یا leaf nodes که لیبل های کلاس ها را نگه میدارند

- Internal nodes (non-leaf nodes) denote a test on an attribute.
- Branches represent outcomes of tests.
- Leaf nodes (terminal nodes) hold class labels.
- Root node is the topmost node.

سوال خوب:
کدوم اتریبیوت را به عنوان ریشه یا روت انتخاب کنیم؟

Apply Model to Test Data

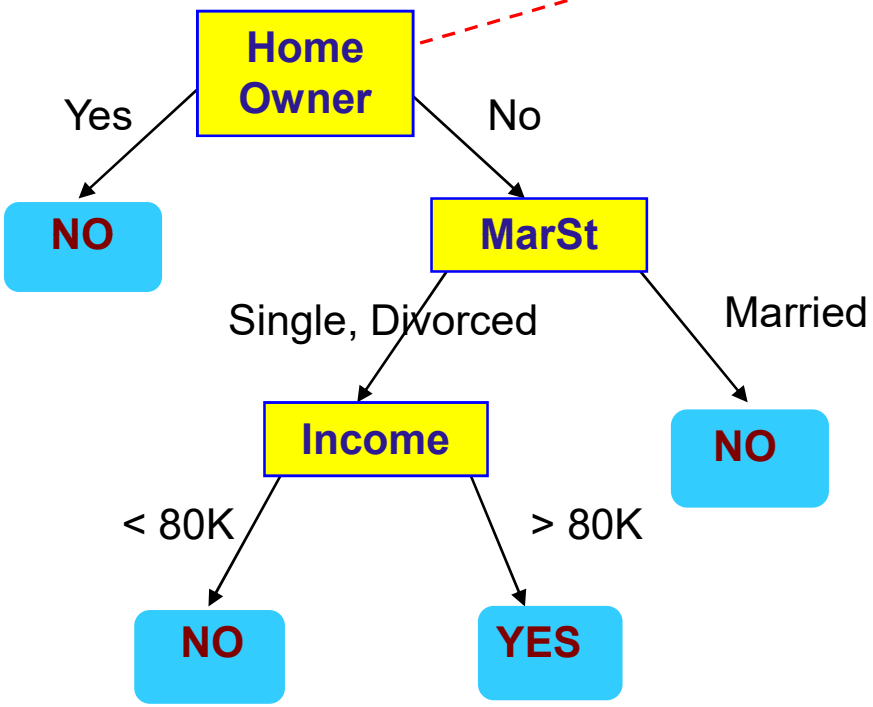


Test Data

Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?

وقتی کلسیفیکیشن میکنیم لزوما به یک مدل نمیرسیم به یک درخت یکسان
لزوما نمیرسیم ینی میتوانیم چندین مدل داشته باشیم روی داده هامون
سوال ۱: چطوری این درخت را بدست اورد؟
چطوری تقسیم بندی کرد شاخه ها را ؟

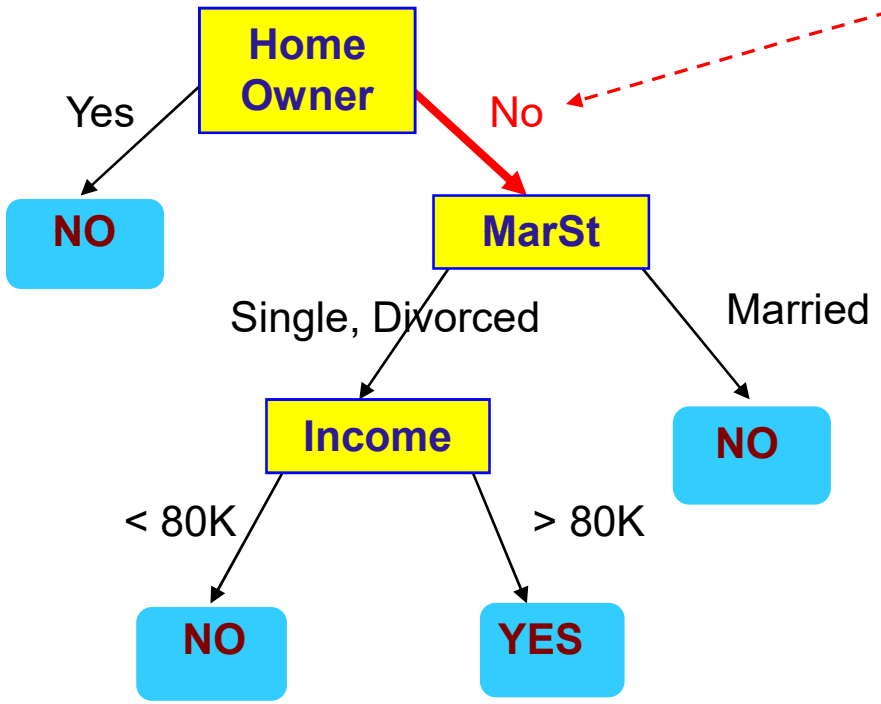
Apply Model to Test Data



Test Data

Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?

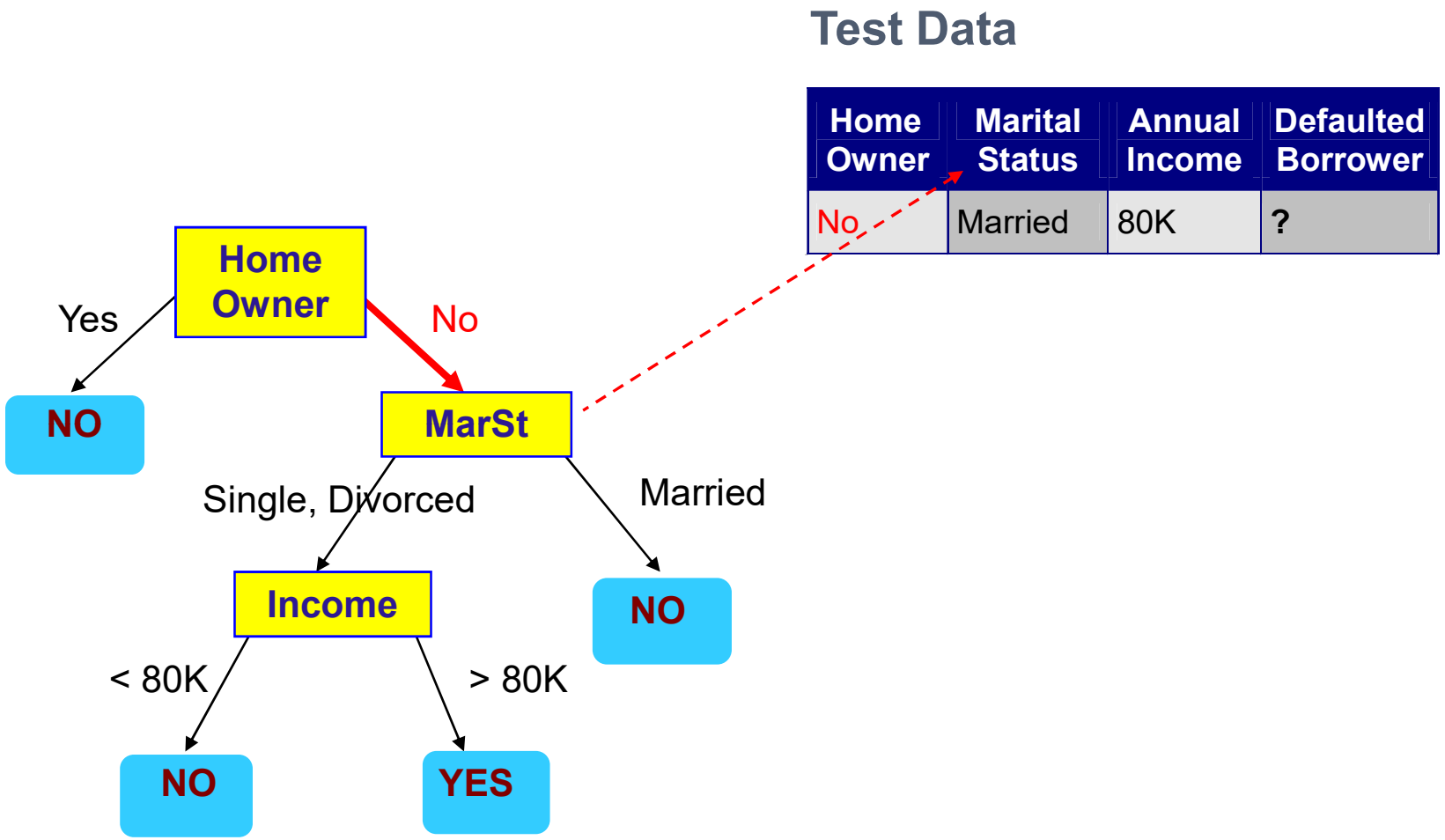
Apply Model to Test Data



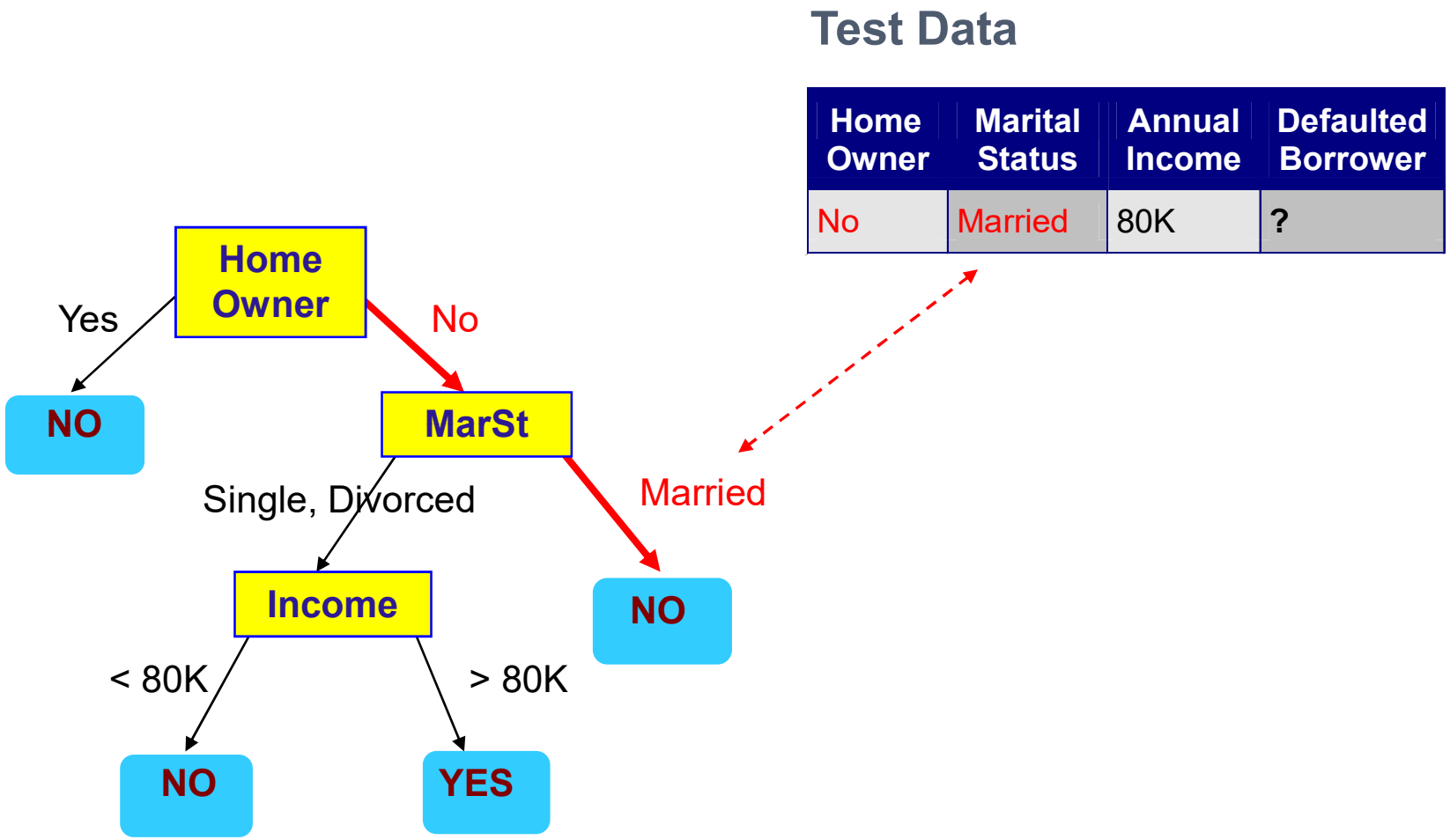
Test Data

Home Owner	Marital Status	Annual Income	Defaulted Borrower
No	Married	80K	?

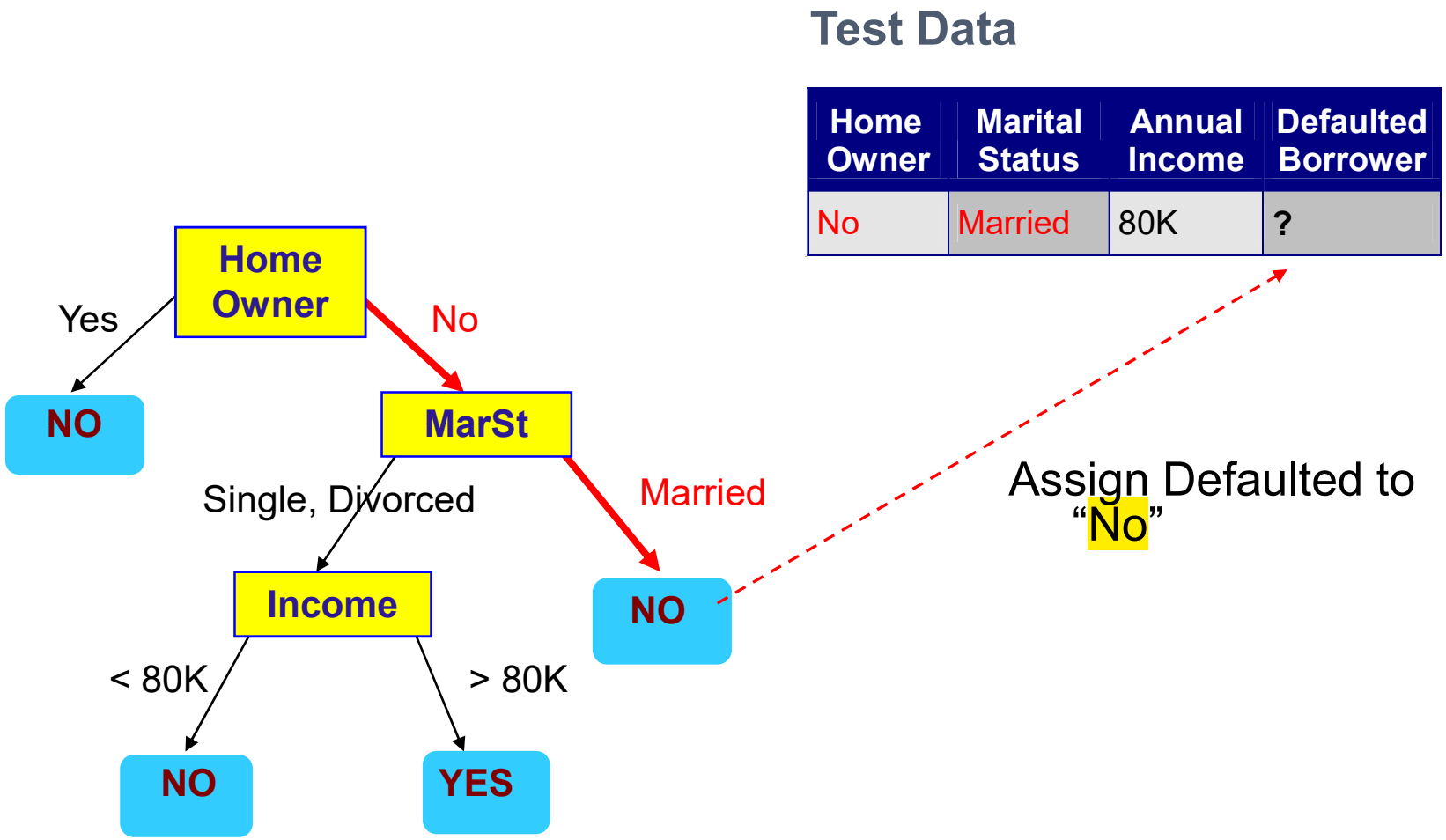
Apply Model to Test Data



Apply Model to Test Data



Apply Model to Test Data



Building an optimal decision tree for your dataset requires careful consideration of several factors. Here are some steps you can follow to build the best decision tree:

Data Preparation: Ensure that your data is properly preprocessed and cleaned. This includes removing missing values, handling outliers, and transforming categorical variables.

Feature Selection: Choose the most relevant features for your model. You can use techniques like correlation analysis, principal component analysis (PCA), or recursive feature elimination (RFE) to identify the most important features.

Splitting Criteria: Select the appropriate splitting criteria to partition your data into the subsets that contain the most homogeneous class labels. Commonly used splitting criteria include Gini index and entropy.

Pruning: Decide when to stop growing the tree by setting a stopping criterion to avoid overfitting. Pruning can improve the accuracy of the decision tree by removing irrelevant or redundant branches.

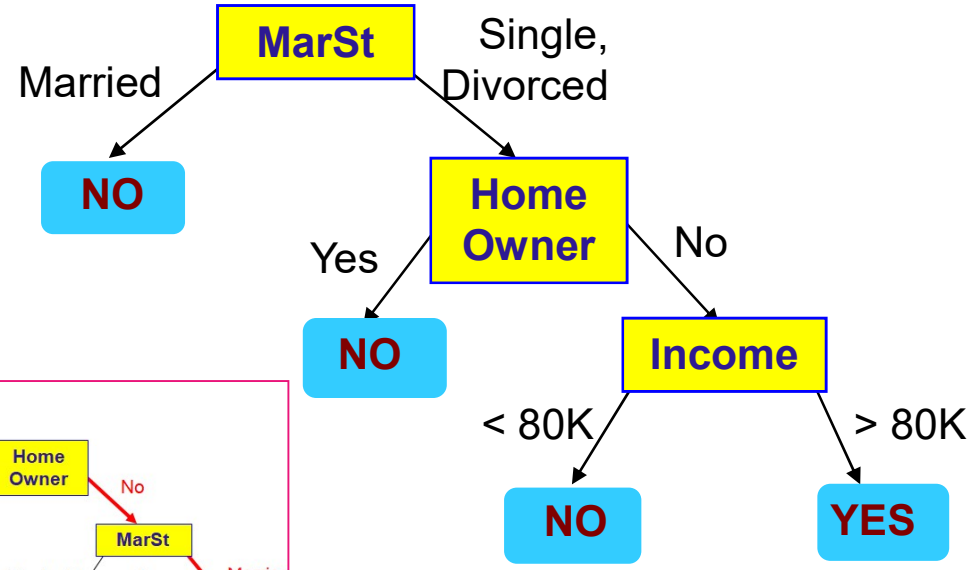
Tuning Parameters: Fine-tune the hyperparameters of your decision tree algorithm. This includes parameters like the maximum depth of the tree, minimum samples required to split a node, and minimum samples required in a leaf.

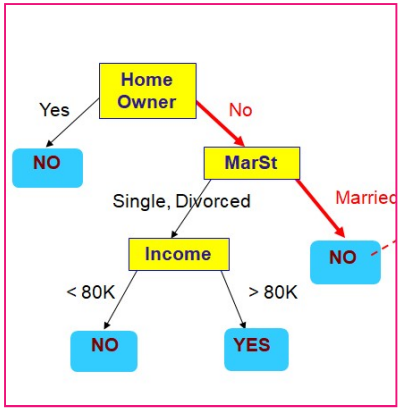
Validation: Evaluate the performance of your decision tree using techniques like cross-validation, holdout validation, or bootstrapping.

By following these steps, you can build the best possible decision tree for your dataset.

Another Example of Decision Tree

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





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

Decision Tree Induction

- Many Algorithms:
  - Hunt's Algorithm (one of the earliest)
  - CART
  - ID3, C4.5
  - SLIQ,SPRINT

General Structure of Hunt's Algorithm

- Let D_t be the set of training records that reach a node t

- General Procedure:
  - If D_t contains records that belong the same class y_t, then

    t is a leaf node labeled as y_t
  - If D_t contains records that belong to more than one class,

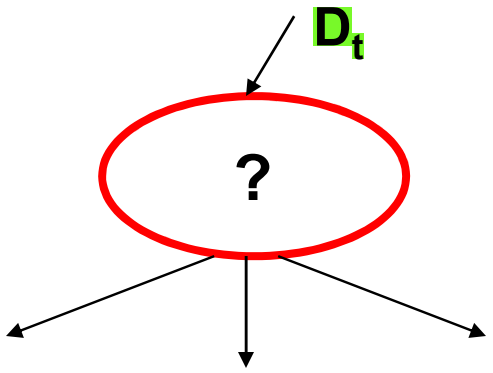
    use an attribute test to split the data into smaller subsets.

    Recursively apply the procedure to each subset.

این نود تی را یکی از این اتریبیوت ها بگذار بعد براساس مقادیر مختلفی که اون اتریبیوت میگیره داده ها وارد شاخه های مختلفی میشن که باعث شکستن مساله به بخش های کوچک تر میشه توی هرشاخه میشه جداگانه برای بحث مدل سازی داده ها اقدام کنیم وقتی که مساله های کوچک تر را حل کردیم مسائل بزرگتر هم حل میشن

اتریبیوتی که انتخاب میکنیم مساله را برامون میشکنه

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



Hunt's Algorithm

Defaulted = No

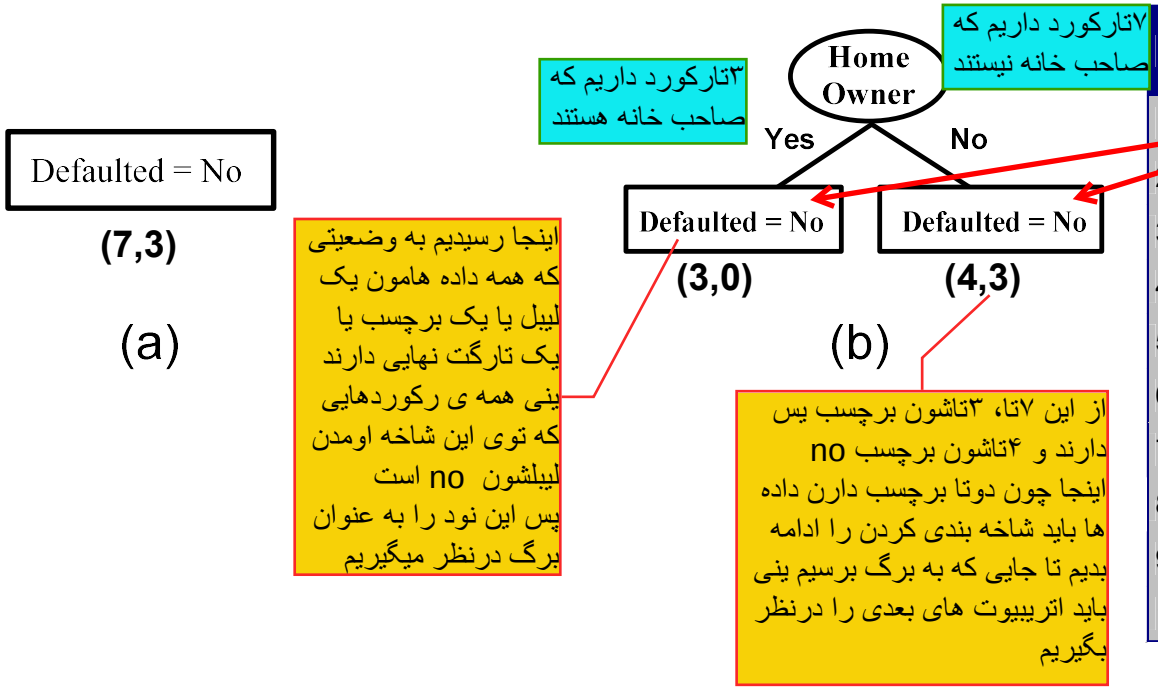
(7,3)

(a)

۱۰تا رکورد داریم که ۳تاشون لیبل yes دارند و
۷ تاشون لیبل no
پس ۷تا برای یک کلاسه و ۳تا برای کلاس دیگه

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

Hunt's Algorithm

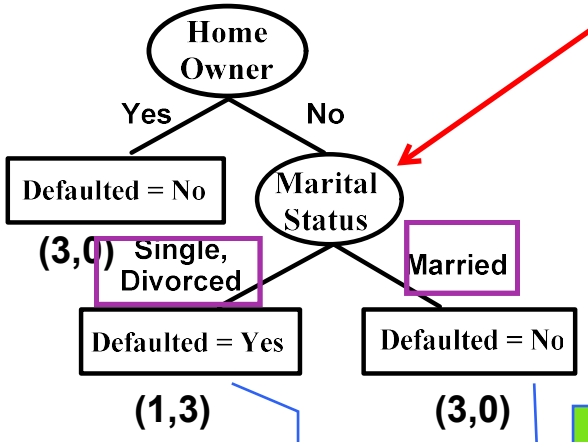


ID	Home Owner	Marital Status	Annual Income	Defaulted Borrower
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Hunt's Algorithm

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

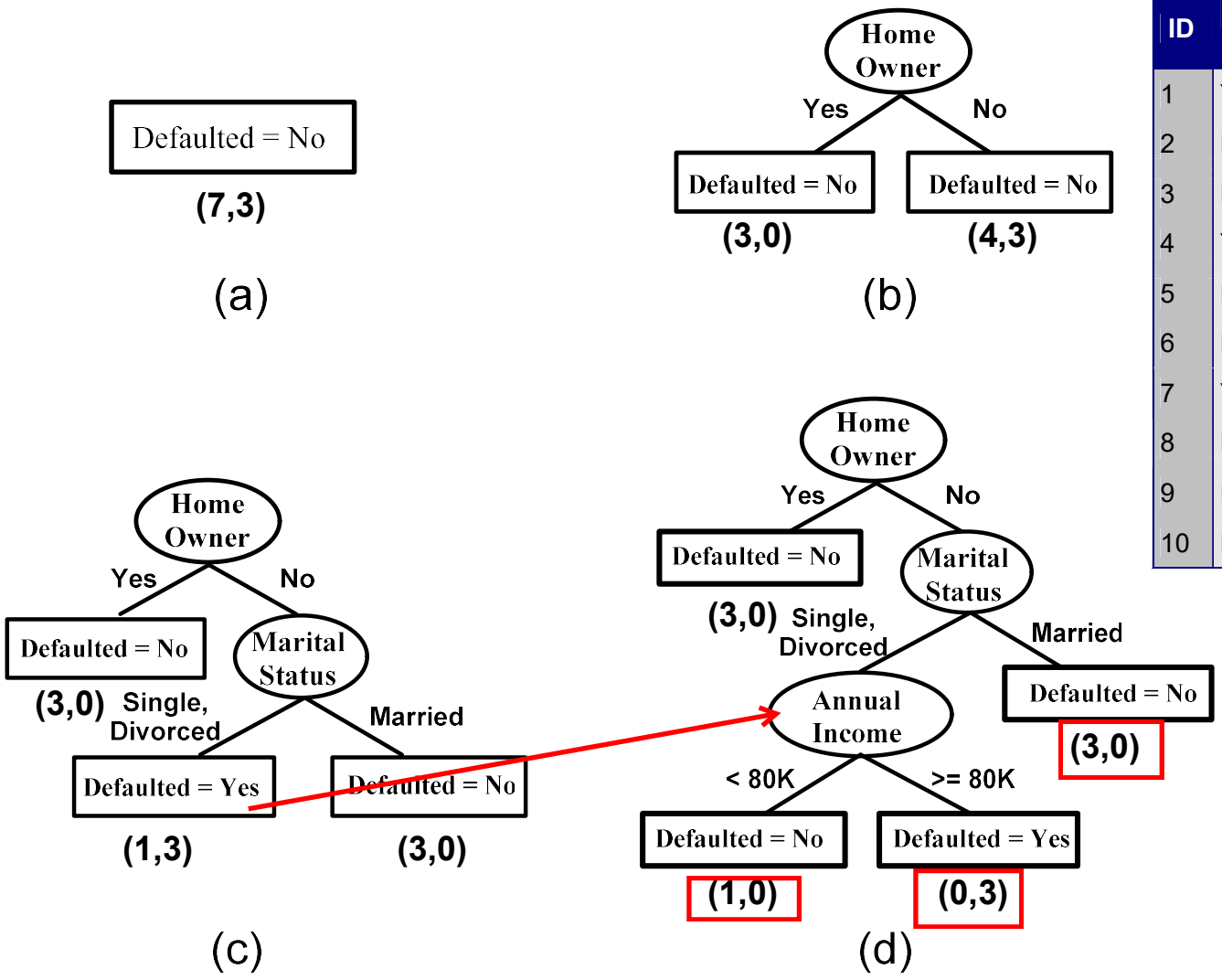
میاییم وضعیت تاهل را براشون بررسی میکنیم



اینجا هنوز عدم قطعیت داریم و برای اینکه بتونیم تصمیم گیری کنیم برای داده ها که درنهایت بشون برچسب یس بدیم یا نه باید براساس یک اتریبیوت دیگه هم تقسیمبندی کنیم

با تقسیم بندی براساس وضعیت ازدواج باعث شد یه شاخه ایجاد بشه که توش همه ی داده ها وضعیت لیبلشون یکسان است یعنی الان سه تا رکورد داریم که همشون لیبل no را دارند پس به یه برگ رسیدیم

Hunt's Algorithm



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

الان قطعیت همه کامل شد ینی همه حالت ها به برگ رسید

Design Issues of Decision Tree Induction

- How should training records be split?
  (splitting criterion)

1. Gini index
2. Entropy
3. information gain

برچه اساسی باید تقسیم بندی کنیم؟
معیار تفکیک کردن

- How should the splitting procedure stop?
  (stopping criterion)

این روند تقسیم کردن و شاخه بندی چه زمانی باید تمام شه؟
معیار توقف

Design Issues of Decision Tree Induction

- How should the splitting procedure stop? (stopping criterion)
  - Stop splitting if all the records belong to the same class or have identical attribute values
  - There are no remaining attributes for further partitioning
  - There are no samples left – majority voting on the parent's samples is employed.

به شاخه ای برسیم که برچسب همه یکسان بشه

جایی متوقف بشیم که هیچ اتریبیوت دیگه ای نیست

یه راگیری باید میگرفتیم که مثلا اونایی که تعداد لیبل یس بیشتر دارند شاخه به اونا تعلق داره

Design Issues of Decision Tree Induction

- How should training records be split? (splitting criterion)
  - Method for expressing test condition
    - depending on attribute types
  - Measure for evaluating the goodness of a test condition

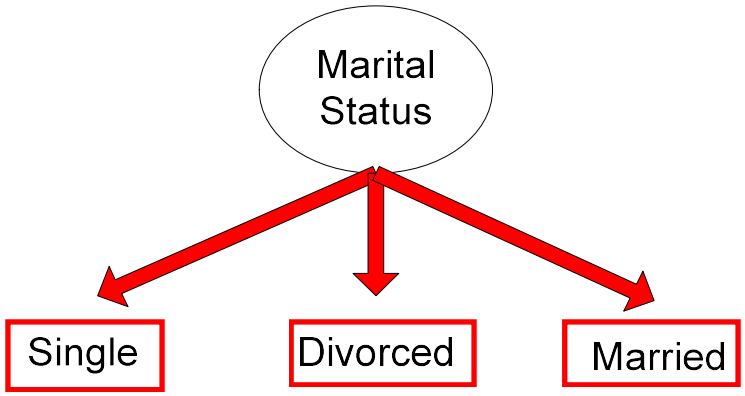
سوال:
عدد ۸۰ هزار را ازکجا ورد برای درامد که براساس اون اومد اسپلیت کرد؟
یا ازکجا فهمید که باید مجرد هارا بفرسته توی یه شاخه و بقیه که شامل متاهل و طلاق گرفته بودن رو بفرسته توی شاخه ی دیگه؟

Methods for Expressing Test Conditions

- Depends on attribute types
  - Binary
  - Nominal
  - Ordinal
  - Continuous

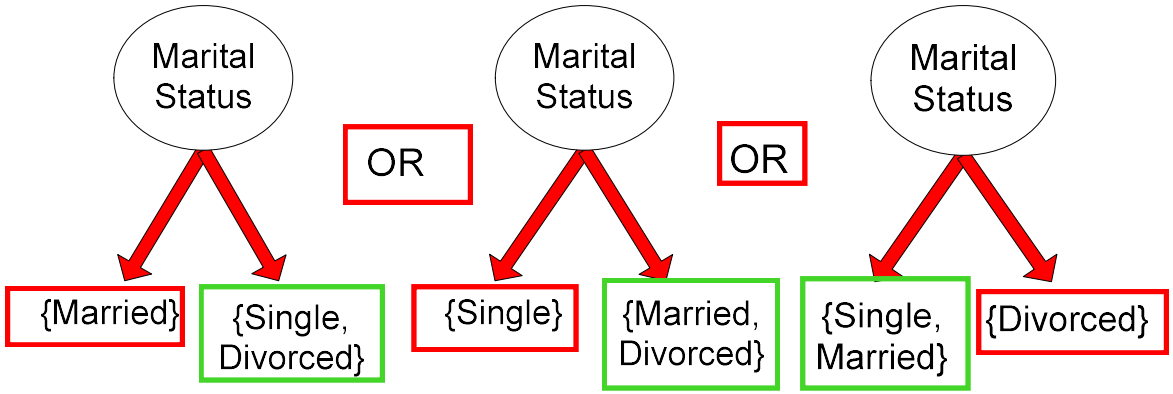
Test Condition for Nominal Attributes

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



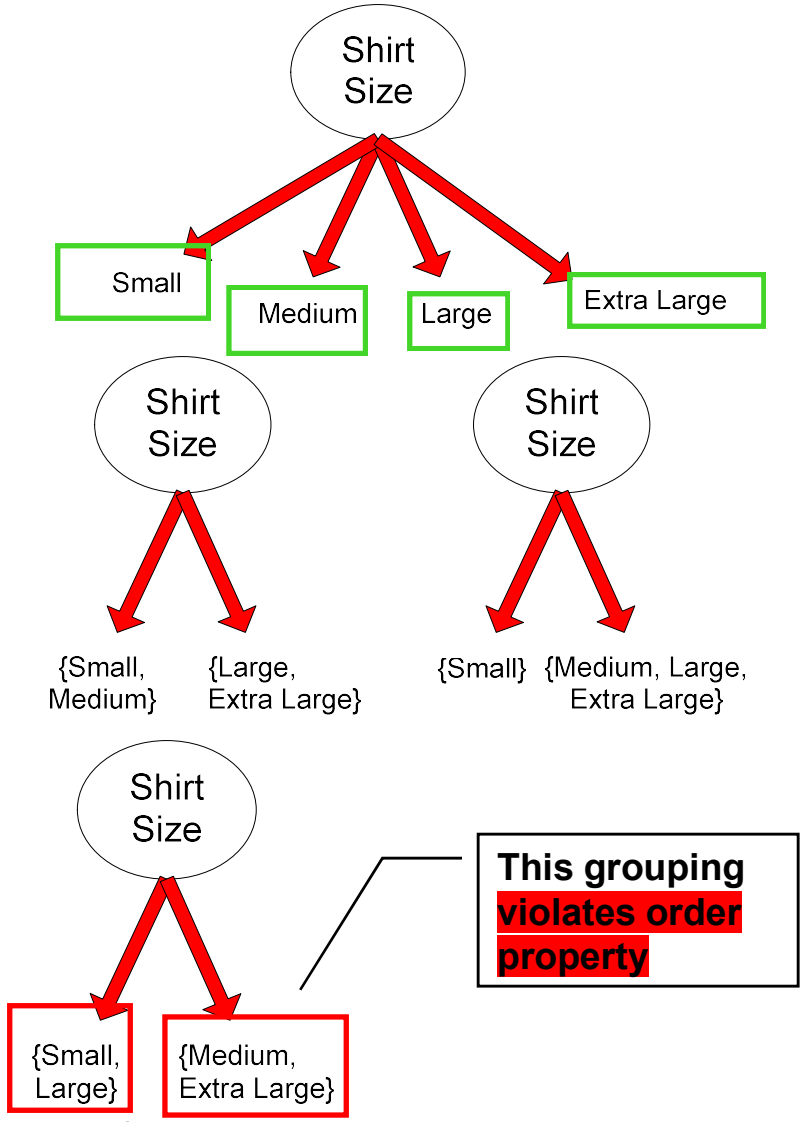
- Binary split:
  - Divides values into two subsets

معیار تفکیک کردن
علاوه بر بستگی داشتن به نوع اتریبیوت به خود اتریبیوت هم بستگی داره اینکه چه اتریبیوتی را انتخاب کنیم برای اسپلیت کردن؟ کدوم را بگذاریم برای نود روت؟ بعدش کدوم انتخاب شه؟
نحوه ی انتخاب کردن مقادیر برای جدا کردن مثل اینکه برای درامد ۸۰ را بگذاریم یا ۸۱؟ متاهل ها با مجرد ها باشن یا با طلاق گرفته ها؟
مساله دوم اینکه کدوم اتریبیوت را زودتر بگذاریم؟

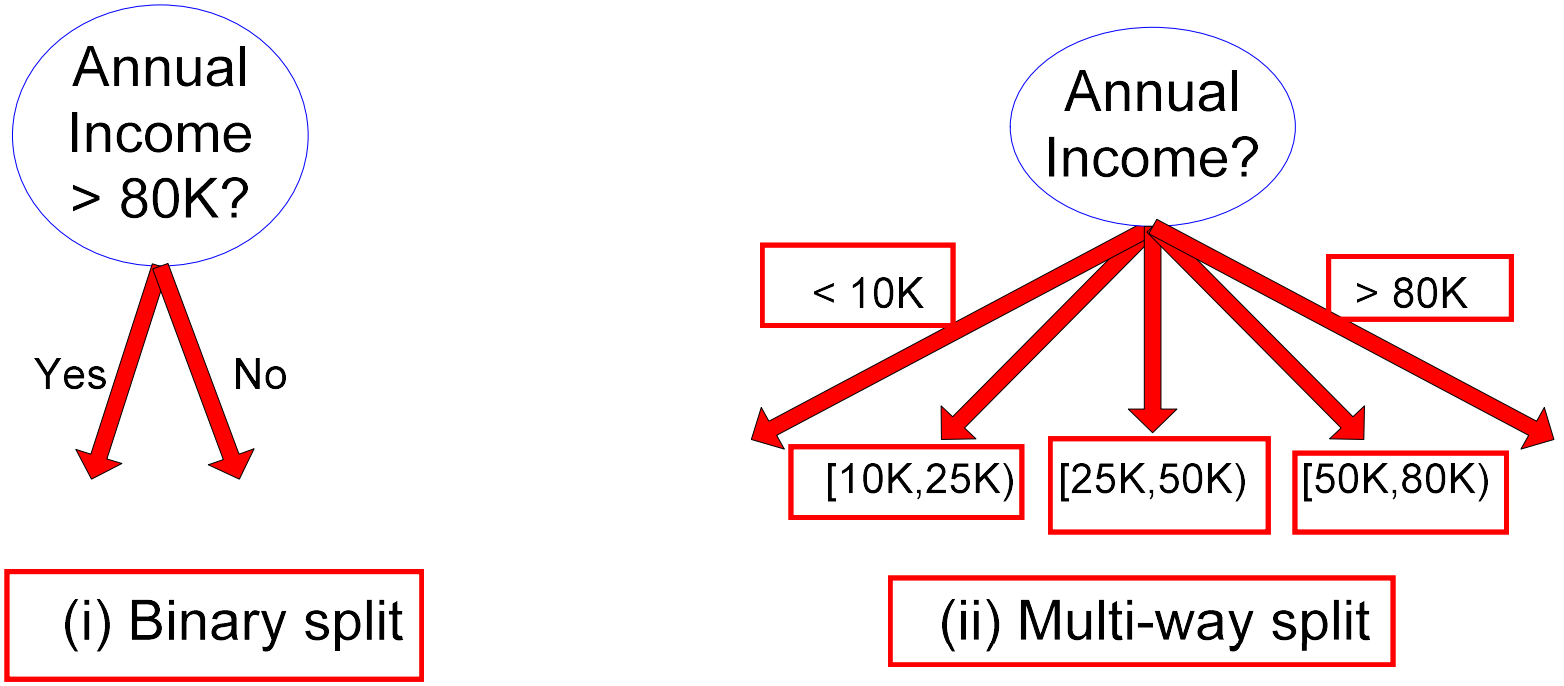


Test Condition for Ordinal Attributes

- Multi-way split:
  - Use as many partitions as distinct values
- Binary split:
  - Divides values into two subsets
  - Preserve order property among attribute values



Test Condition for Continuous Attributes



مثل درامد و وزن

Splitting Based on Continuous Attributes

- Different ways of handling
  - Discretization to form an ordinal categorical attribute

    Ranges can be found by equal interval bucketing, equal frequency bucketing (percentiles), or clustering.

    بازه بندی کردن مقادیر

    - Static – discretize once at the beginning
    - Dynamic – repeat at each node

  - Binary Decision: $(A < v)$ or $(A \geq v)$
    - consider all possible splits and finds the best cut
    - can be more compute intensive

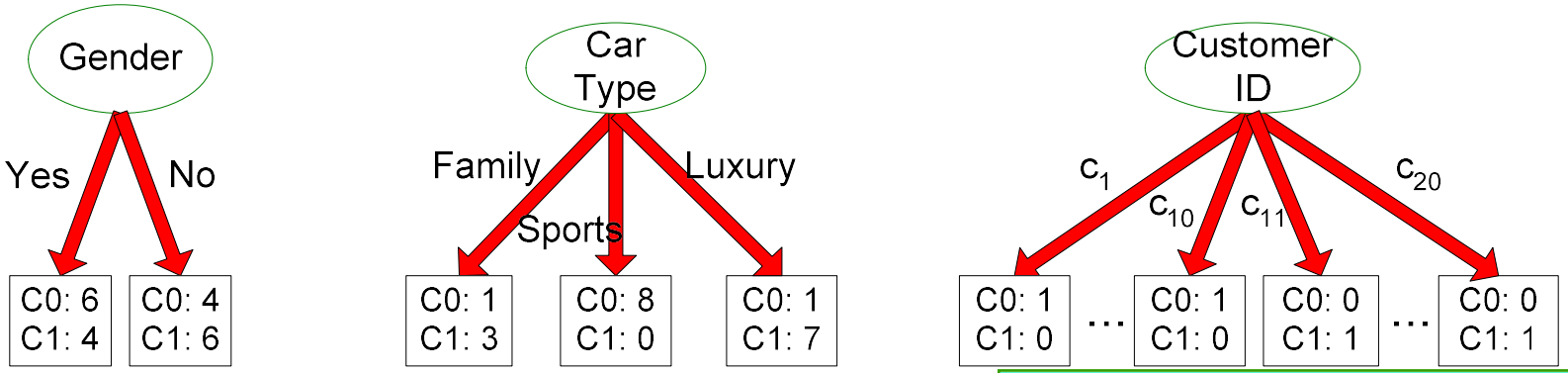
How to determine the Best Split

۲۰تا رکورد داریم و دوتا کلاس
۱۰تا از رکوردها برای کلاس
صفر است و ۱۰تا برای یک

Customer Id	Gender	Car Type	Shirt Size	Class
1	M	Family	Small	C0
2	M	Sports	Medium	C0
3	M	Sports	Medium	C0
4	M	Sports	Large	C0
5	M	Sports	Extra Large	C0
6	M	Sports	Extra Large	C0
7	F	Sports	Small	C0
8	F	Sports	Small	C0
9	F	Sports	Medium	C0
10	F	Luxury	Large	C0
11	M	Family	Large	C1
12	M	Family	Extra Large	C1
13	M	Family	Medium	C1
14	M	Luxury	Extra Large	C1
15	F	Luxury	Small	C1
16	F	Luxury	Small	C1
17	F	Luxury	Medium	C1
18	F	Luxury	Medium	C1
19	F	Luxury	Medium	C1
20	F	Luxury	Large	C1

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

۳تاانتخاب داریم
برای نود روت
۱.جنسیت بشه
۲.نوع ماشین
۳.ایدی مشتری



Which test condition is the best?

بیشتر از همه خالص و
پیور بکنه داده ها را

یه معیاری میخایم که به تصمیم گیریمون کمک کنه مثلا تقسیم بندی را پیورتر بکنه مثلا اینطوری نباشه که اول کار داده ها مون ۵۰ ۵۰ باشن و بعد از تقسیم بندی هم هنوز اون ابهامه توی تقسیم بندیمون باشه اینطوری هیچی از تقسیم بندی نفهمیدیم و انگار اصلا تقسیم بندی نکردیم پس دنبال یه معیاری هستیم که میزان خالص بودن تقسیم بندی را اندازه بگیره و هرکدوم خالص تر بود انتخاب کنیم

اندازه گیری میزان خالص شدن به کمک معیار های زیر

Measures of Node Impurity

- Gini Index

به ازای کل کلاس هایی که توی نود داریم.

$$Gini\ Index = 1 - \sum_{i=0}^{c-1} p_i(t)^2$$

Where $p_i(t)$ is the frequency of class i at node t, and c is the total number of classes

احتمال هرکلاسی که داریم در یک نود مشخص مثل t

- Entropy

$$Entropy = -\sum_{i=0}^{c-1} p_i(t) log_2 p_i(t)$$

- Misclassification error

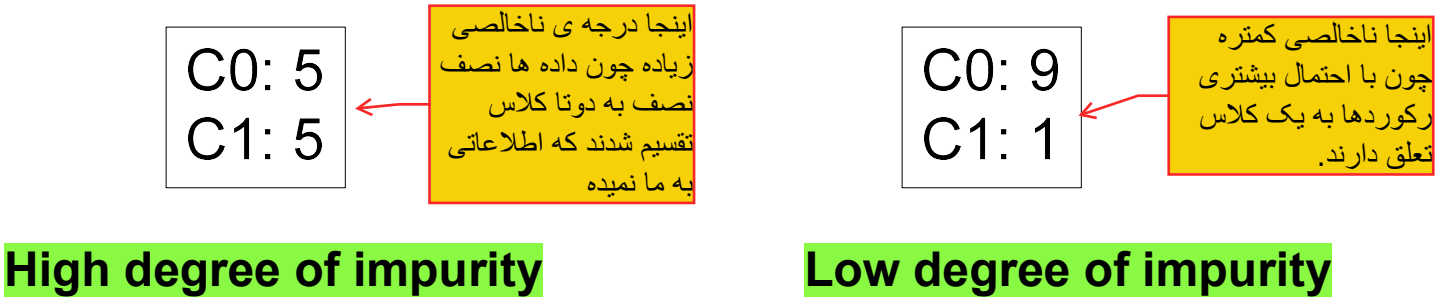
$$Classification\ error = 1 - \max[p_i(t)]$$

ماکس احتمال رخدادن یک کلاس توی یک نود

How to determine the Best Split

- Greedy approach:
  - Nodes with purer class distribution are preferred

- Need a measure of node impurity:



Finding the Best Split

1. Compute impurity measure (P) before splitting
2. Compute impurity measure (M) after splitting
   - Compute impurity measure of each child node
   - M is the weighted impurity of child nodes
3. Choose the attribute test condition that produces the highest gain

Gain = P - M

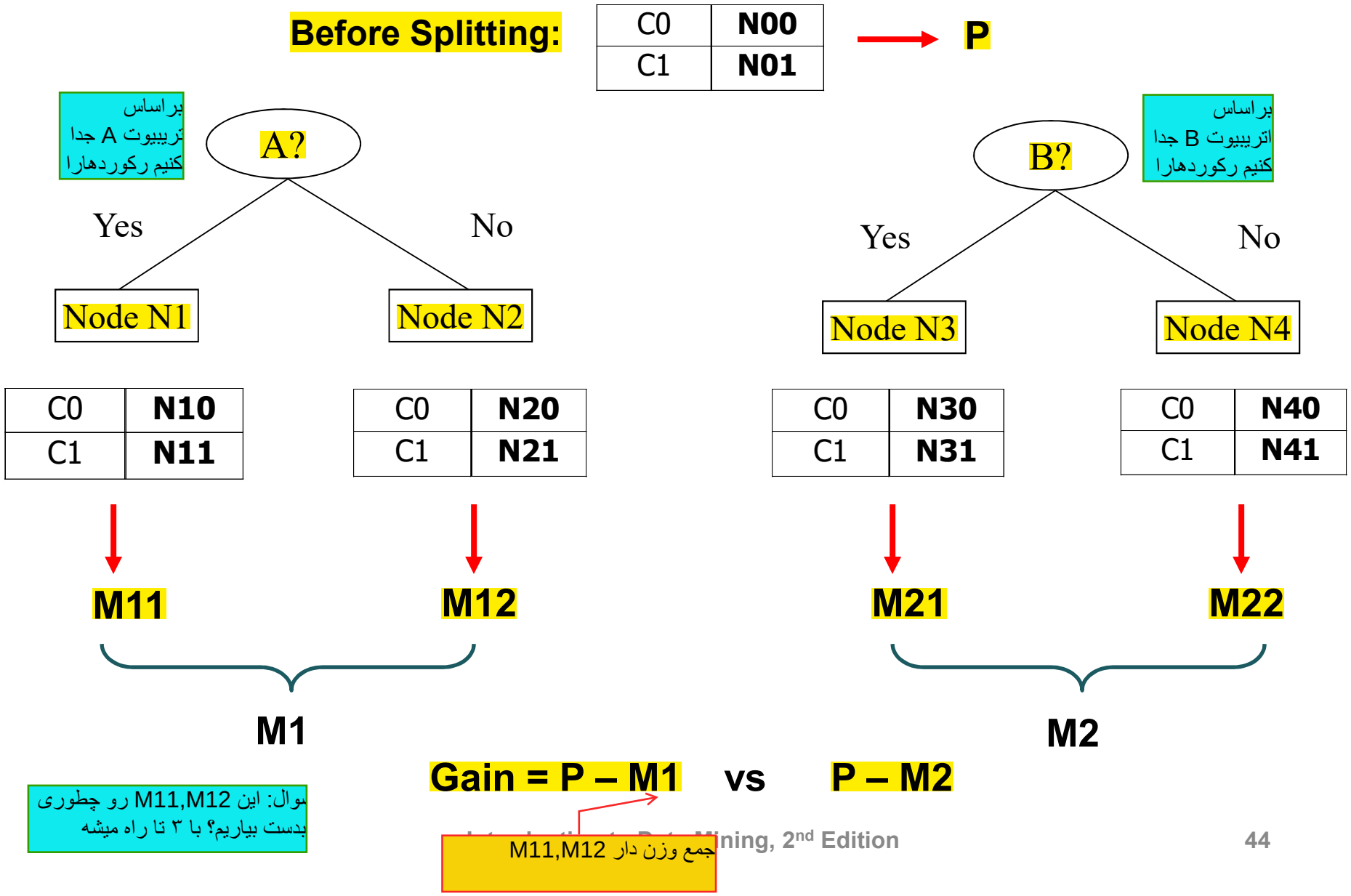
میزانی که از ناخالصی داده هامون کم شده میشه گین ما یا دستاورد ما هرچی این میزان بیشتر باشه بهتره

بهترین split اونیه که
۱. بعد از انجام دادنش، بیشترین گین رو بدست بیاریم
یا میشه بگیم
۲. بعد از اسپلیت شدن، کمترین میزان ناخالصی را بمون بده

or equivalently, lowest impurity measure after splitting (M)

اونی که کمترین M را میده انتخاب میکنیم.

Finding the Best Split



این معیار جینی ایندکس داره ناخالصی رو نشون میده ما هم که به دنبال خالص شدن تقسیم بندی هامون هستیم پس بهترین حالتش اینه که جینی ایندکس صفر بشه یعنی ناخالصی نداشته باشیم و بدترین حالتش اینه که احتمال همه ی کلاس ها توی یک نود یکسان بشه یعنی توزیع یکسان بین کلاس ها

Measure of Impurity: GINI

- Gini Index for a given node t

اگه احتمال همه کلاس ها برابر باشه:
c * (1/c)^2 =
c *(1/c ^2) = 1/c
gini index = 1-1/c

$$Gini\ Index = 1 - \sum_{i=0}^{c-1} p_i(t)^2$$

Where $p_i(t)$ is the frequency of class i at node t, and c is the total number of classes

  - Maximum of $1 - 1/c$ when records are equally distributed among all classes, implying the least beneficial situation for classification
  - Minimum of 0 when all records belong to one class, implying the most beneficial situation for classification

ااگه احتمال یکی از کلاس ها یک باشه و بقیه صفر باشه :
gini index = 0
ینی قطعیت داریم و مثلا برچسب همه ی دیتاهامون یه چیز یکسان شده

Measure of Impurity: GINI

- Gini Index for a given node t :

$$Gini\ Index = 1 - \sum_{i=0}^{c-1} p_i(t)^2$$

  - For 2-class problem (p, 1 – p):

binary classification

GINI = $1 - p^2 - (1 - p)^2 = 2p\ (1\text{-}p)$

C1	0
C2	6
Gini=0.000	

بهترین حالت که ناخالصی صفر است.

C1	1
C2	5
Gini=0.278	

C1	2
C2	4
Gini=0.444	

C1	3
C2	3
Gini=0.500	

بدترین حالت که رکوردها به صورت مساوی بین کلاس ها توزیع شدند.
p(c1) = p(c2) = 3/6 = 1/2
gini index = 1 - 1/c
c =2
چون ۲تاکلاس داریم پس
gini index = 1 - 1/2 = 1/2

Computing Gini Index of a Single Node

$$Gini\ Index = 1 - \sum_{i=0}^{c-1} p_i(t)^2$$

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

Computing Gini Index for a Collection of Nodes

- When a node p is split into k partitions (children)

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

جینی هر پارتیشن را در احتمال اون تعداد رکوردی که رفتند توی اون پارتیشن ضرب میکنیم و باهم جمع میکنیم

where, n_i = number of records at child i,

n = number of records at parent node p.

جع وزن دار جینی ایندکس ها

Example 1:
Suppose we have a dataset with 100 samples and two classes (A and B). We want to split the data based on a feature X that can take on two values (0 or 1). The Gini index for the original dataset is:

Gini(D) = 1 - (50/100)^2 - (50/100)^2 = 0.5

Now, let's say that when X=0, there are 40 samples of class A and 10 samples of class B, and when X=1, there are 10 samples of class A and 40 samples of class B. The Gini index for each subset is:

Gini(D_0) = 1 - (40/50)^2 - (10/50)^2 = 0.32
Gini(D_1) = 1 - (10/50)^2 - (40/50)^2 = 0.32

To calculate the Gini index for the split, we take a weighted average of the Gini indices for each subset:

Gini_split = (50/100)*Gini(D_0) + (50/100)*Gini(D_1) = 0.32

If we compare this to the original Gini index of 0.5, we can see that the split has reduced the impurity of the data.

Example 2:
Suppose we have a dataset with 100 samples and three classes (A, B, and C). We want to split the data based on a feature X that can take on three values (0, 1, or 2). The Gini index for the original dataset is:

Gini(D) = 1 - (30/100)^2 - (40/100)^2 - (30/100)^2 = 0.66

Now, let's say that when X=0, there are 20 samples of class A, 5 samples of class B, and 5 samples of class C. When X=1, there are 5 samples of class A, 20 samples of class B, and 5 samples of class C. When X=2, there are 5 samples of class A, 5 samples of class B, and 20 samples of class C. The Gini index for each subset is:

Gini(D_0) = 1 - (20/30)^2 - (5/30)^2 - (5/30)^2 = 0.53
Gini(D_1) = 1 - (5/30)^2 - (20/30)^2 - (5/30)^2 = 0.53
Gini(D_2) = 1 - (5/30)^2 - (5/30)^2 - (20/30)^2 = 0.53

To calculate the Gini index for the split, we take a weighted average of the Gini indices for each subset:

Gini_split = (30/100)*Gini(D_0) + (30/100)*Gini(D_1) + (40/100)*Gini(D_2) = 0.53

If we compare this to the original Gini index of 0.66, we can see that the split has reduced the impurity of the data.

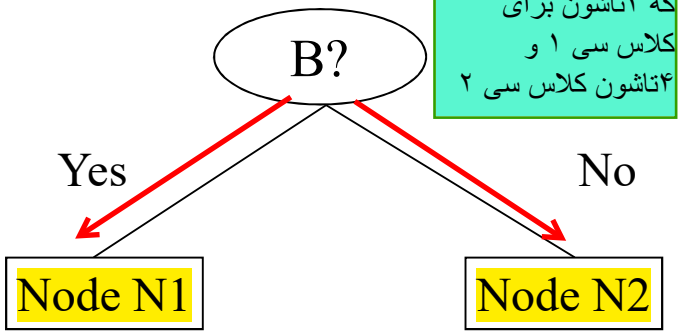
Binary Attributes: Computing GINI Index

- Splits into two partitions (child nodes)
- Effect of Weighing partitions:
  - Larger and purer partitions are sought

p1 = 7/12
p2=5/12
gini index = 1-((7/12)^2 + (5/12)^2)) =
1-(0.34 + 0.17) = 1-0.51 = 0.49

۶تا اومدن این شاخه که ۱شون برای کلاس سی ۱ و ۵تا برای سی ۲

۶تا هم اومدن این شاخه که ۲تاشون برای کلاس سی ۱ و ۴تاشون کلاس سی ۲



	Parent
C1	7
C2	5
Gini = 0.486	

۱۲ تا داده پدرشون بوده

Gini(N1)
= 1 – (5/6)² – (1/6)²
= 0.278

Gini(N2)
= 1 – (2/6)² – (4/6)²
= 0.444

	N1	N2
C1	5	2
C2	1	4
Gini=0.361		

Weighted Gini of N1 N2
= 6/12 * 0.278 +
6/12 * 0.444
= 0.361

محاسبه ی حالت وزن دار

Gain = 0.486 – 0.361 = 0.125

این مقدار بمون کمک میکنه بین یه اتریبیوت و اتریبیوت های دیگه مقایسه کنیم

Categorical Attributes: Computing Gini Index

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

۳تا فرزند درست میشه از این اسپلیت کردن پس ۳تا جینی خواهیم داشت که باید در احتمال پارتیشن بندی شاخه ها هم ضرب کنیم که بشه کل جینی این اسپلیت

Multi-way split

	CarType		
	Family	Sports	Luxury
C1	1	8	1
C2	3	0	7
Gini	0.163		

Two-way split
(find best partition of values)

	CarType	
	{Sports, Luxury}	{Family}
C1	9	1
C2	7	3
Gini	0.468	

	CarType	
	{Sports}	{Family, Luxury}
C1	8	2
C2	0	10
Gini	0.167	

Which of these is the best?

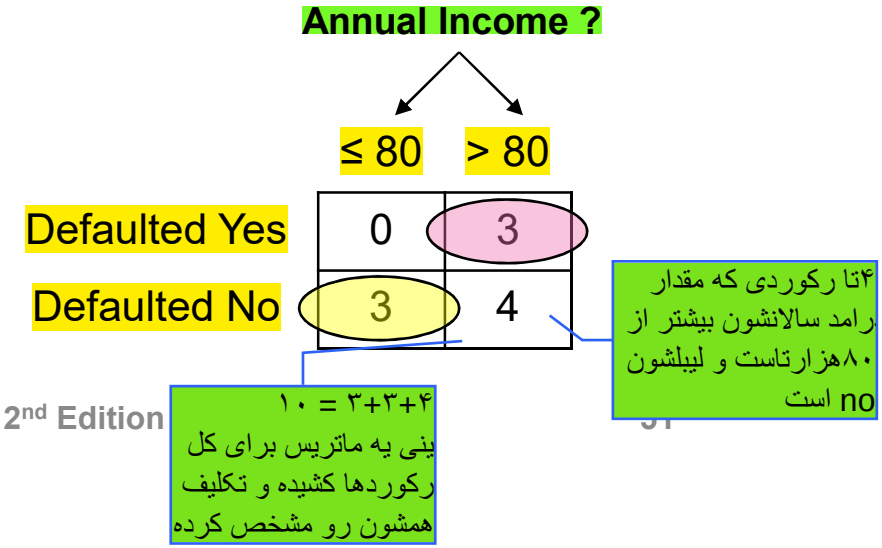
از بین این ۳تا حالت اونی که کمترین جینی را میده انتخاب میکنیم.یعنی حالت multi way split

جینی تک تک این حالت هارو بدست میاریم، جینی پرنت یا والد هم که داریم اونی را انتخاب میکنیم که بیشترین گین را بده بمون

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.

ID	Home Owner	Marital Status	Annual Income	Defaulted
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



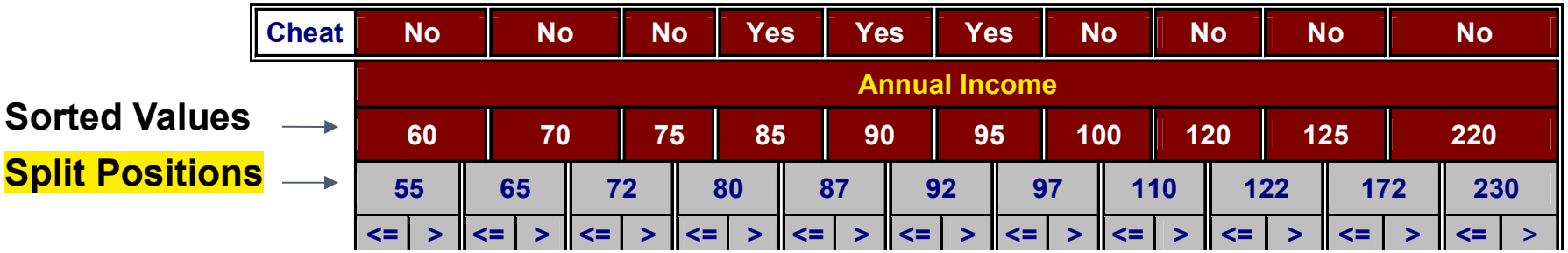
Continuous Attributes: Computing Gini Index...

- 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
Annual Income										
Sorted Values →	60	70	75	85	90	95	100	120	125	220

Continuous Attributes: Computing Gini Index...

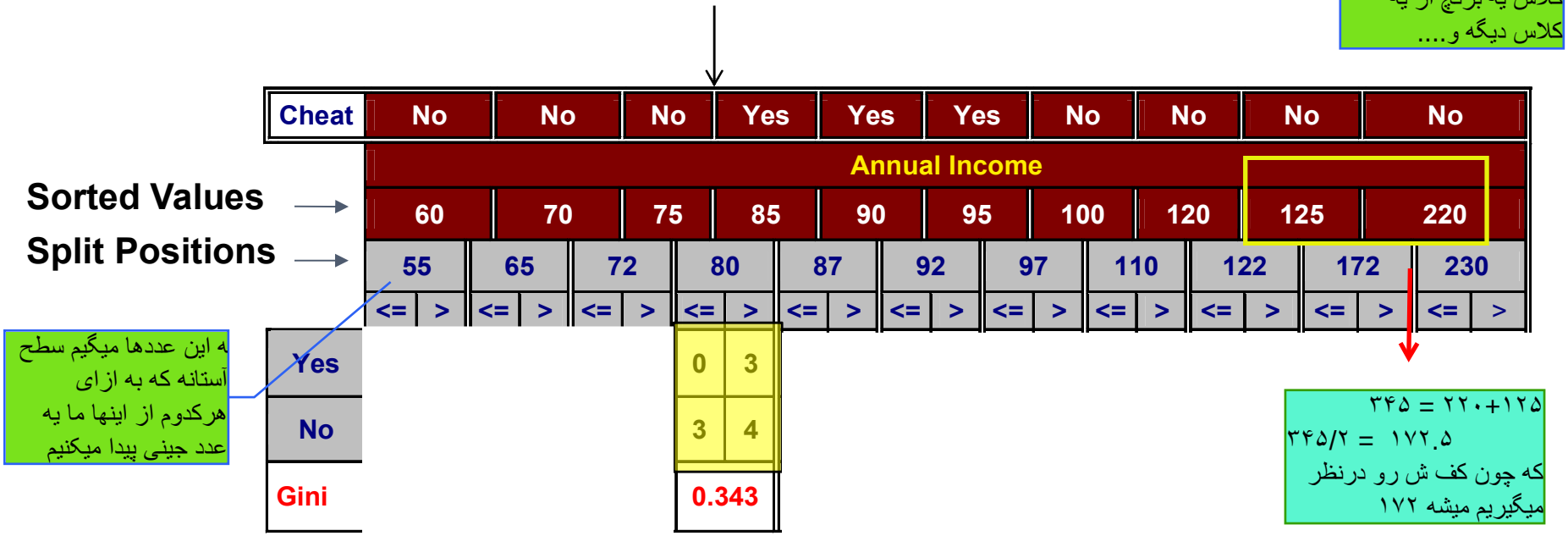
- For efficient computation: for each attribute,
  - Sort the attribute on values
  - Linearly scan these values, each time updating the count matrix and computing gini index
  - Choose the split position that has the least gini index



Continuous Attributes: Computing Gini Index...

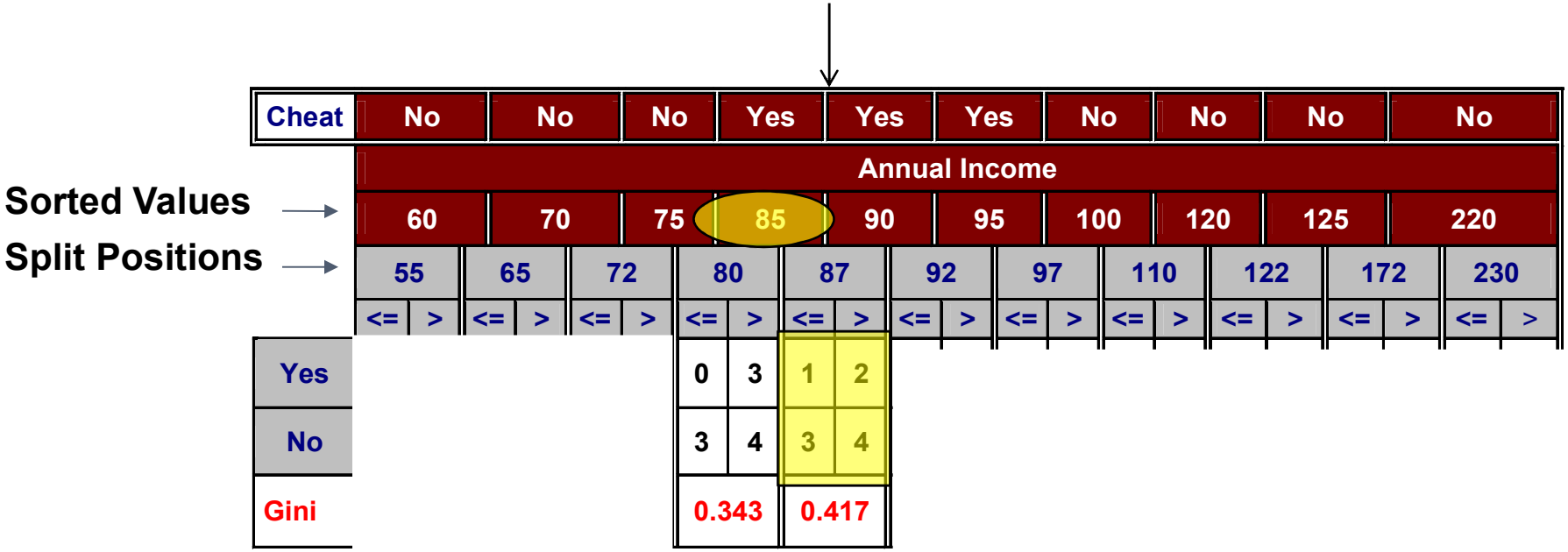
- For efficient computation: for each attribute,
  - Sort the attribute on values
  - Linearly scan these values, each time updating the count matrix and computing gini index
  - Choose the split position that has the least gini index

کم کردن عدم قطعیت افزایش قطعیت مثلا یه برنچ بشه واقعا از یه کلاس یه برنچ از یه کلاس دیگه و....



Continuous Attributes: Computing Gini Index...

- 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



Continuous Attributes: Computing Gini Index...

- 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	
Annual 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	

Measure of Impurity: Entropy

راجع به عدم قطعیت داریم حرف میزنیم فزایش انتروپی به معنای فزایش عدم قطعیت است

- Entropy at a given node t

$$Entropy = -\sum_{i=0}^{c-1} p_i(t) log_2 p_i(t)$$

Where $p_i(t)$ is the frequency of class i at node t, and c is the total number of classes

  - Maximum of $\log_2 c$ when records are equally distributed among all classes, implying the least beneficial situation for classification
  - Minimum of 0 when all records belong to one class, implying most beneficial situation for classification

- Entropy based computations are quite similar to the GINI index computations

Computing Entropy of a Single Node

$$Entropy = -\sum_{i=0}^{c-1} p_i(t) log_2 p_i(t)$$

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

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

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

فزایش انتروپی که معادل افزایش عدم قطعیت است و ما دوست نداریم :)

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

Computing Information Gain After Splitting

نود پرنت: انتروپی قبل از اینکه این اتریبیوت را انتخاب کنیم چقدر بوده؟

آنتروپی چقدر کاهش پیداکرده؟ یعنی چقدر به قطعیت رسیدیم بعد از تقسیم بندی براساس اون اتریبیوت
ما میخایم هرچی در درخت پایین تر میریم قطعیتمون بیشتر بشه دیگه

- Information Gain:

$$Gain_{split} = Entropy(p) - \sum_{i=1}^{k} \frac{n_i}{n} Entropy(i)$$

آنتروپی بعد از انتخاب نود برای تصمیم گیری چون با چندتا مجموعه سروکار داریم باید وزن هرشاخه را حساب کنیم

Parent Node, p is split into k partitions (children)

n_i is number of records in child node i

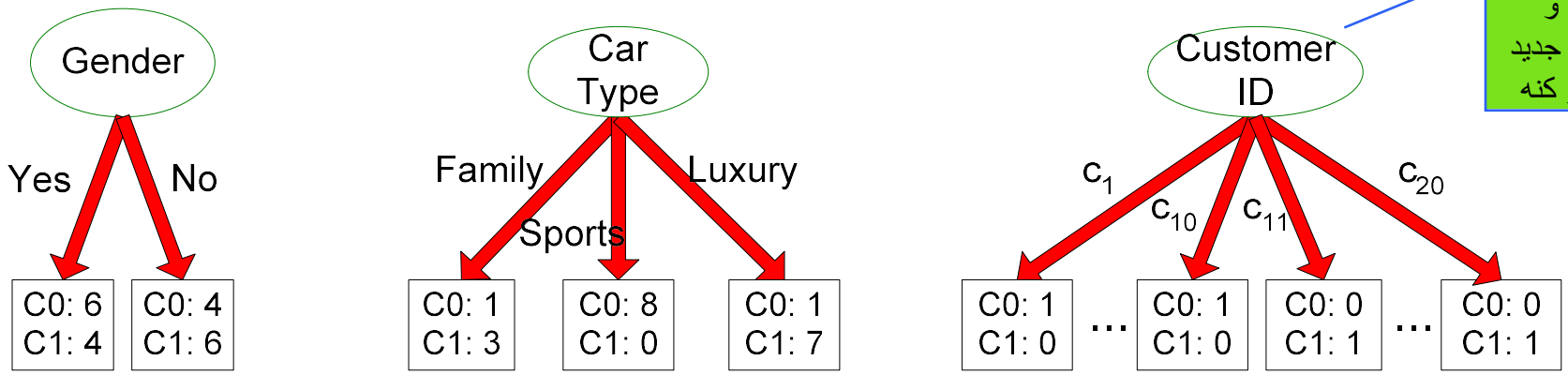
  - Choose the split that achieves most reduction (maximizes GAIN)
  - Used in the ID3 and C4.5 decision tree algorithms
  - Information gain is the mutual information between the class variable and the splitting variable

مثلا ما بادانستن شماره دانشجویی یه نفری نمیتونیم درباره ی ویژگی های دیگه ش حرف بزنیم !! با انتخاب آیدی ما بیشتر حفظ کردیم ویژگی ها را و هیچ یادگیری اتفاق نیفتاده

Problem with large number of partitions

- Node impurity measures tend to prefer splits that result in large number of partitions, each being small but pure

کاستمر ایدی که نتخاب میشه اول کار باعث میشه مدل هیچی یادنگیره و برای داده های جدید ندانه که چیکار کنه



  - Customer ID has highest information gain because entropy for all the children is zero

مشکل اینکه ایدی رو به عنوان اتریبیوت انتخاب کنیم چیه؟ داده ی جدید که بیاد نمیتونیم توی یه دسته بگذاریمش چون ایدیش فرق داره میخاد تو یه دسته جدید بره نه دسته های قبلی پس به دسته بندی داده ها کمک نمیکنه

ینی عدم قطعیت داریم چون هر شاخه توش یه رکورد قرار گرفته

زمانی که بااتریبیوت هایی مثل تاریخ و آیدی سروکار داریم

Gain Ratio

برای حل شدن مشکل انتخاب زودهنگام آیدی به عنوان اتریبیوت تقسیم بندی در درخت، باید یه جریمه ای براش بگذاریم

- Gain Ratio:

یه شاخصی از تعداد شاخه ها اضافه میکنیم

اتریبیوت هایی که باعث میشه داده ها به تعداد زیادی از شاخه ها افراز بشن و در هر شاخه یه تعداد برابری کلاس داریم

$$Gain\ Ratio = \frac{Gain_{split}}{Split\ Info}$$

یه نسبتی از گین است

$$Split\ Info = -\sum_{i=1}^{k} \frac{n_i}{n} log_2 \frac{n_i}{n}$$

برای کاستمر آیدی مقدارش بزرگ میشه و باعث کم شدن مقدار گین کلی میشه پنی جریمه میکنه

داره انتروپی شاخه رو حساب میکنه

Parent Node, p is split into k partitions (children)

n_i is number of records in child node i

- Adjusts Information Gain by the entropy of the partitioning ($Split\ Info$).
  - Higher entropy partitioning (large number of small partitions) is penalized!

جریمه میشه گین کلی

- Used in C4.5 algorithm
- Designed to overcome the disadvantage of Information Gain

Gain Ratio

- Gain Ratio:

$$Gain\ Ratio = \frac{Gain_{split}}{Split\ Info} \qquad Split\ Info = \sum_{i=1}^{k} \frac{n_i}{n} log_2 \frac{n_i}{n}$$

Parent Node, p is split into k partitions (children)

n_i is number of records in child node i

	CarType		
	Family	Sports	Luxury
C1	1	8	1
C2	3	0	7
Gini	0.163		

SplitINFO = 1.52

	CarType	
	{Sports, Luxury}	{Family}
C1	9	1
C2	7	3
Gini	0.468	

SplitINFO = 0.72

	CarType	
	{Sports}	{Family, Luxury}
C1	8	2
C2	0	10
Gini	0.167	

SplitINFO = 0.97

n1/n = 8/(8+2+10) = 8/20
n2/n = 12/20
8/20 * log 8/20 + 12/20 * log 12/20 =
-0.528 -0.438 = -0.966

Suppose we have a dataset with 100 examples, where 80 examples belong to class A and 20 belong to class B. Let's split this dataset based on an attribute X that has three possible values: X1, X2, and X3. If we calculate the information gain for each value of X, we get:

- For X1, the dataset splits into 60 examples of class A and 20 examples of class B. The information gain is 0.539.
- For X2, the dataset splits into 10 examples of class A and 10 examples of class B. The information gain is 0.208.
- For X3, the dataset splits into 10 examples of class A and 0 examples of class B. The information gain is 0.206.

Now, let's calculate the split information for each value of X based on the number of examples:

- For X1, the split information is 1 log(1) + 3/5 log (3/5) + 2/5 log(2/5) = 0.971.
- For X2, the split information is 1 log(1) + 1 log(1) = 0.
- For X3, the split information is 1 log(1) + 0 log(0) = 0.

Finally, we can calculate the gain ratio for each value of X by dividing the information gain by the split information:

- For X1, the gain ratio is 0.539/0.971 = 0.556.
- For X2, the gain ratio is undefined since the split information is 0.
- For X3, the gain ratio is undefined since the split information contains a log(0).

Therefore, based on the gain ratio criterion, we would choose X1 as the attribute to split the dataset.

Suppose we have a dataset with two binary features (A and B) and a binary target variable (Y), where 10 instances have A=0,B=0, 20 instances have A=1,B=0, 15 instances have A=0,B=1, and 5 instances have A=1,B=1. Let's compute the gain ratio for each possible split:

Split on feature A:

Subgroup 1: A=0 (25 instances)
Y=0: 10 instances
Y=1: 15 instances
Subgroup 2: A=1 (25 instances)
Y=0: 20 instances
Y=1: 5 instances
Entropy(Y) = -(30/50)*log2(30/50) - (20/50)*log2(20/50) = 0.971
Split information = -(25/50)*log2(25/50) - (25/50)*log2(25/50) = 1.0
Information gain = 0.971 - ((25/50)*(-(10/25)*log2(10/25) - (15/25)log2(15/25)) + (25/50)(-(20/25)*log2(20/25) - (5/25)*log2(5/25))) = 0.02
Gain ratio = 0.02 / 1.0 = 0.02
Split on feature B:

Subgroup 1: B=0 (30 instances)
Y=0: 20 instances
Y=1: 10 instances
Subgroup 2: B=1 (20 instances)
Y=0: 10 instances
Y=1: 10 instances
Entropy(Y) = -(30/50)*log2(30/50) - (20/50)*log2(20/50) = 0.971
Split information = -(30/50)*log2(30/50) - (20/50)*log2(20/50) = 0.971
Information gain = 0.971 - ((30/50)*(-(20/30)*log2(20/30) - (10/30)log2(10/30)) + (20/50)(-(10/20)*log2(10/20) - (10/20)*log2(10/20))) = 0.019
Gain ratio = 0.019 / 0.971 = 0.0196
Based on the gain ratios, we would choose to split on feature A because it has a slightly higher gain ratio than feature B.

Suppose we have a dataset with three features (A, B, and C) and a binary target variable (Y), where 15 instances have A=0,B=0,C=0, 10 instances have A=1,B=0,C=0, 5 instances have A=0,B=1,C=0, 5 instances have A=0,B=0, C=1, 5 instances have A=1,B=1,C=0, and 20 instances have A=1,B=0,C=1. Let's compute the gain ratio for each possible split:

Split on feature A:

Subgroup 1: A=0 (25 instances)
Y=0: 15 instances
Y=1: 10 instances
Subgroup 2: A=1 (30 instances)
Y=0: 25 instances
Y=1: 5 instances
Entropy(Y) = -(30/55)*log2(30/55) - (25/55)*log2(25/55) = 0.961
Split information = -(25/55)*log2(25/55) - (30/55)*log2(30/55) = 0.994
Information gain = 0.961 - ((25/55)*(-(15/25)*log2(15/25) - (10/25)log2(10/25)) + (30/55)(-(25/30)*log2(25/30) - (5/30)*log2(5/30))) = 0.132
Gain ratio = 0.132 / 0.994 = 0.133
Split on feature B:

Subgroup 1: B=0 (35 instances)
Y=0: 25 instances
Y=1: 10 instances
Subgroup 2: B=1 (20 instances)
Y=0: 15 instances
Y=1: 5 instances
Entropy(Y) = -(30/55)*log2(30/55) - (25/55)*log2(25/55) = 0.961
Split information = -(35/55)*log2(35/55) - (20/55)*log2(20/55) = 0.928
Information gain = 0.961 - ((35/55)*(-(25/35)*log2(25/35) - (10/35)log2(10/35)) + (20/55)(-(15/20)*log2(15/20) - (5/20)*log2(5/20))) = 0.114
Gain ratio = 0.114 / 0.928 = 0.123
Split on feature C:

Subgroup 1: C=0 (35 instances)
Y=0: 25 instances
Y=1: 10 instances
Subgroup 2: C=1 (20 instances)
Y=0: 15 instances
Y=1: 5 instances
Entropy(Y) = -(30/55)*log2(30/55) - (25/55)*log2(25/55) = 0.961
Split information = -(35/55)*log2(35/55) - (20/55)*log2(20/55) = 0.928
Information gain = 0.961 - ((35/55)*(-(25/35)*log2(25/35) - (10/35)log2(10/35)) + (20/55)(-(15/20)*log2(15/20) - (5/20)*log2(5/20))) = 0.114
Gain ratio = 0.114 / 0.928 = 0.123
Based on the gain ratios, we would choose to split on feature A because it has the highest gain ratio.

Sure, here are some numeric examples of splitting decision trees based on entropy:

Suppose we have a dataset with two binary features (A and B) and a binary target variable (Y), where 10 instances have A=0,B=0, 20 instances have A=1,B=0, 15 instances have A=0,B=1, and 5 instances have A=1,B=1. Let's compute the entropy for each possible split:

- Split on feature A:
  - Subgroup 1: A=0 (25 instances)
    - Y=0: 10 instances
    - Y=1: 15 instances
  - Subgroup 2: A=1 (25 instances)
    - Y=0: 20 instances
    - Y=1: 5 instances
  - Entropy(Y) = -(30/50)*log2(30/50) - (20/50)*log2(20/50) = 0.971
  - Information gain = Entropy(Y) - ((25/50)*(-(10/25)*log2(10/25) - (15/25)*log2(15/25)) + (25/50)*(-(20/25)*log2(20/25) - (5/25)*log2(5/25))) = 0.321 - 0.02 = 0.301

- Split on feature B:
  - Subgroup 1: B=0 (30 instances)
    - Y=0: 20 instances
    - Y=1: 10 instances
  - Subgroup 2: B=1 (20 instances)
    - Y=0: 10 instances
    - Y=1: 10 instances
  - Entropy(Y) = -(30/50)*log2(30/50) - (20/50)*log2(20/50) = 0.971
  - Information gain = Entropy(Y) - ((30/50)*(-(20/30)*log2(20/30) - (10/30)*log2(10/30)) + (20/50)*(-(10/20)*log2(10/20) - (10/20)*log2(10/20))) = 0.321 - 0.019 = 0.302

Based on the information gains, we would choose to split on feature B because it has a slightly higher information gain than feature A.

Note that for entropy-based splitting, the choice of split can be affected by the size of the subgroups, while this is not an issue for gain ratio-based splitting.

Measure of Impurity: Classification Error

- Classification error at a node t

$$Error(t) = 1 - \max_i [p_i(t)]$$

  - Maximum of $1 - 1/c$ when records are equally distributed among all classes, implying the least interesting situation
  - Minimum of 0 when all records belong to one class, implying the most interesting situation

Computing Error of a Single Node

$$Error(t) = 1 - \max_i [p_i(t)]$$

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**

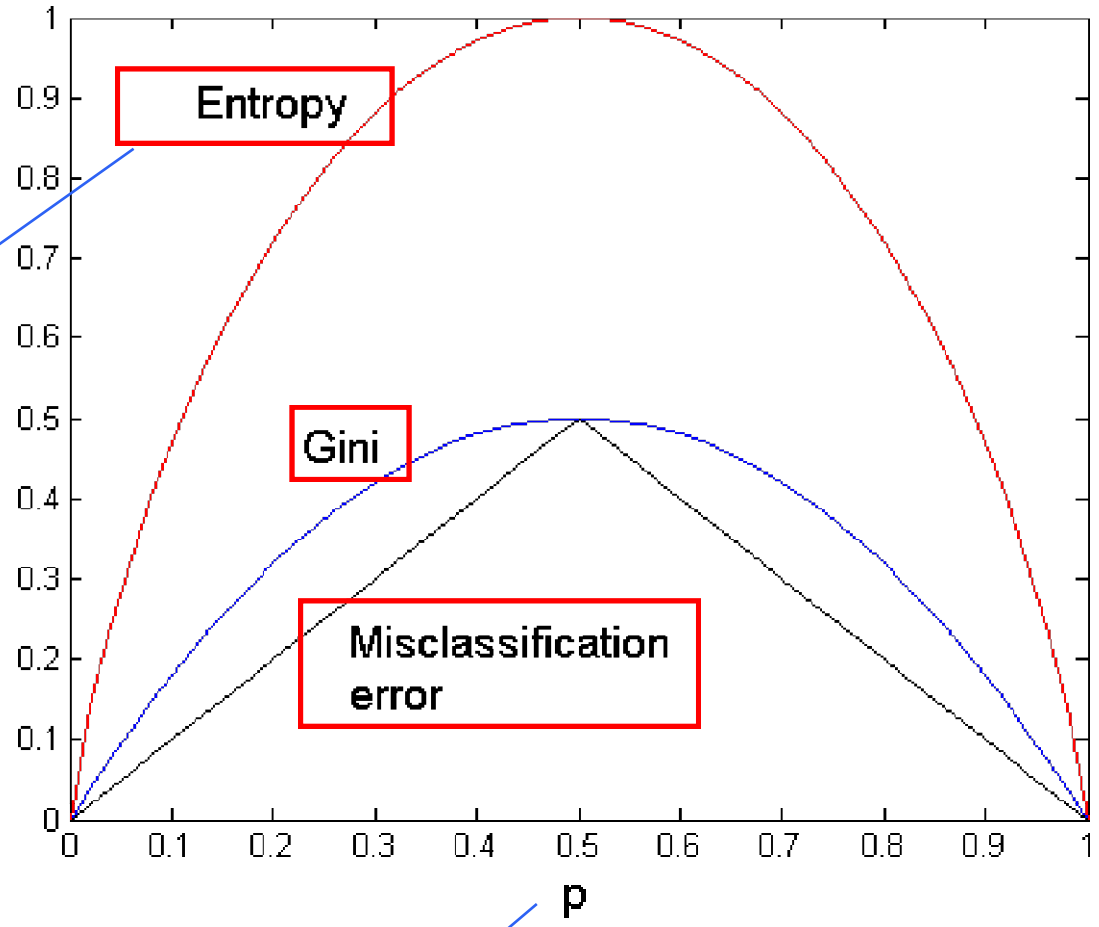
P(C1) = 2/6 P(C2) = 4/6

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

ارور ناشی از افزایش عدم قطعیت اره زیادتر میشه

Comparison among Impurity Measures

For a 2-class problem:



آنتروپی بهتره چون حساسیتش بیشتره برخلاف روش misclassificatio n error که رفتار خطی داره،انتروپی رفتار دقیق تری داره

مقایسه ی شاخص ها براساس احتمال های متفاوت

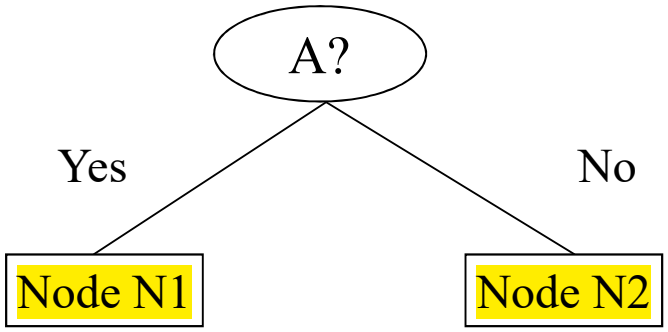
1. Gini index:
The Gini index is a measure of the impurity of a set of instances. It is defined as the probability of misclassifying a random instance in the dataset if it were randomly labeled according to the class distribution in the dataset. The Gini index ranges from 0 to 1, where 0 indicates a pure node (all instances belong to the same class) and 1 indicates a completely impure node (instances are equally distributed among all classes). The Gini index is faster to compute than entropy and is preferred when the class distribution is balanced.

2. Entropy:
Entropy is a measure of the impurity of a set of instances. It is defined as the sum of the negative logarithms of the class probabilities. The entropy ranges from 0 to 1, where 0 indicates a pure node and 1 indicates a completely impure node. Entropy is slower to compute than the Gini index but is preferred when the class distribution is imbalanced.

3. Classification error:
Classification error is a measure of the impurity of a set of instances. It is defined as the proportion of instances in a node that do not belong to the majority class. The classification error ranges from 0 to 1, where 0 indicates a pure node and 1 indicates a completely impure node. Classification error is the simplest impurity measure to compute but is less sensitive to changes in the class distribution than the Gini index and entropy.

In summary, the choice of impurity measure depends on the characteristics of the dataset, such as the class distribution and the size of the dataset. The Gini index is preferred when the class distribution is balanced, while entropy is preferred when the class distribution is imbalanced. Classification error is the simplest impurity measure to compute but is less sensitive to changes in the class distribution than the other measures.

Misclassification Error vs Gini Index



	Parent
C1	7
C2	3
Gini = 0.42	

Gini(N1)
= 1 – $(3/3)^2$ – $(0/3)^2$
= 0

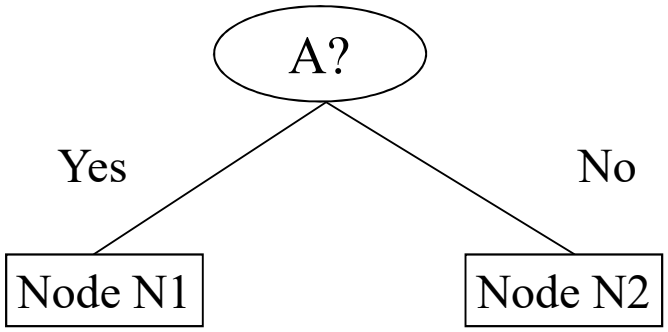
Gini(N2)
= 1 – $(4/7)^2$ – $(3/7)^2$
= 0.489

	N1	N2
C1	3	4
C2	0	3
Gini=0.342		

Gini(Children)
= 3/10 * 0
+ 7/10 * 0.489
= 0.342

Gini improves but error remains the same!!

Misclassification Error vs Gini Index



	Parent
C1	**7**
C2	**3**
Gini = 0.42	

	N1	N2
C1	**3**	**4**
C2	**0**	**3**
Gini=0.342		

	N1	N2
C1	**3**	**4**
C2	**1**	**2**
Gini=0.416		

Misclassification error for all three cases = 0.3 !

Decision Tree Based Classification

- Advantages:
  - Relatively inexpensive to construct
  - Extremely fast at classifying unknown records
  - Easy to interpret for small-sized trees
  - Robust to noise (especially when methods to avoid overfitting are employed)
  - Can easily handle redundant attributes
  - Can easily handle irrelevant attributes (unless the attributes are interacting)

شناسایی اتریبیوت های نامرتبط

یه سری اتریبیوت هستند که برای تصمیم گیری براشون باید باهم بشون نگاه کرد جداجدا که نگاه کنیم نمیشه تصمیم گرفت براشون به اینا interacting میگن attributes

- Disadvantages: .
  - Due to the greedy nature of splitting criterion, interacting attributes (that can distinguish between classes together but not individually) may be passed over in favor of other attributed that are less discriminating.
  - Each decision boundary involves only a single attribute

ویژگی های خوب درخت های تصمیم:
۱. خیلی راحت ساخته میشوند
۲. خیلی سریع آموزش میبینند
۳. تفسیر پذیر هستند به خصوص درخت های کوچکتر و خصوصا برای افرادی که با ماشین لرنینگ اشنایی ندارند
۴.نسبت به نویز نسبتا رباست و مقاوم است چون با خود رکوردها کار نمیکنیم و بااحتمال هاشون کار میکنیم
۵. جاهایی که اتریبیوت های اضافه داریم میتونه بگه این اضافه است و کاری بش نداشته باش به دلیل شباهت رفتارهای اتریبیوت ها پس برای شناسایی اتریبیوت های اضافه مفید است

how decision trees can handle redundant attributes?
Decision trees can handle redundant attributes in different ways. One common approach is to use feature selection techniques to identify and remove redundant attributes before building the tree. This can help reduce the complexity of the tree and improve its accuracy.

Another approach is to use pruning techniques after building the tree to eliminate redundant branches or subtrees that do not contribute significantly to the classification performance. Pruning can help simplify the tree and reduce overfitting, which may occur when the tree is too complex and captures noise or irrelevant features in the data.

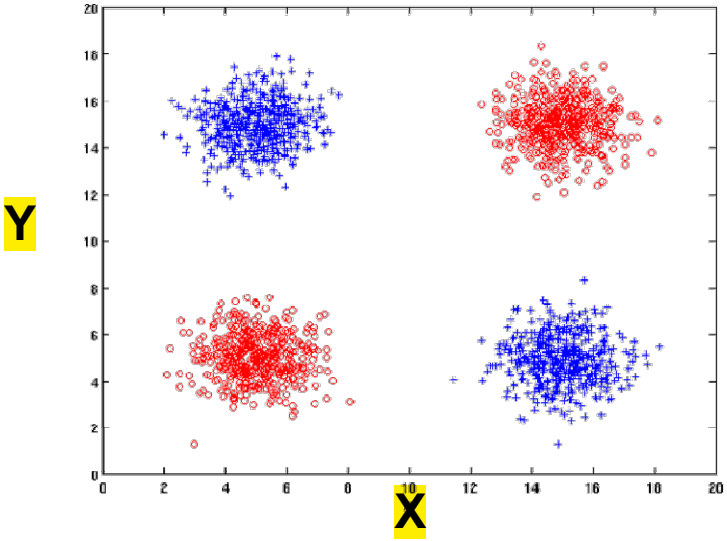
Moreover, some decision tree algorithms such as C4.5 or CART use a measure of information gain or impurity reduction to select the best split at each node. These measures take into account both the relevance and redundancy of the attributes, and aim to find splits that maximize the overall accuracy of the tree while minimizing redundancy.

Single attribute-based decision boundaries refer to the use of a single feature or attribute to make decisions in a decision tree. In other words, at each node in the tree, only one feature is used to split the data into two or more groups based on some threshold value.

For example, consider a decision tree that predicts whether or not a customer will buy a product based on their age. The first node of the tree may split the data into two groups: customers younger than a certain age and customers older than that age. This is an example of a single attribute-based decision boundary.

Single attribute-based decision boundaries are commonly used in decision trees because they are easy to interpret and visualize. However, as I mentioned earlier, they have some limitations in terms of their ability to capture complex relationships and handle missing data.

Handling interactions



+ : 1000 instances

o : 1000 instances

Entropy (X) : 0.99

Entropy (Y) : 0.99

Disadvantages:

Decision trees are prone to overfitting, which means the model may perform well on the training data but poorly on the test data.
Small changes in the data can lead to large changes in the decision tree.
Decision trees can be biased towards features with many levels or features that provide more information.
They are not suitable for learning complex patterns or relationships in data.
Decision trees can be unstable because small variations in the data can result in a completely different tree.

Let's say we have a dataset that includes information about customers at a car dealership. The dataset includes the following attributes:

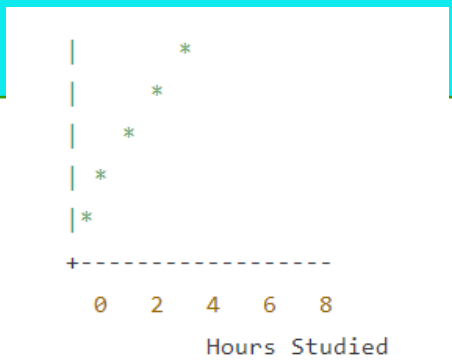
Age: the age of the customer in years
Income: the annual income of the customer in thousands of dollars
Gender: the gender of the customer (either male or female)
Car type: the type of car the customer is interested in (e.g., sedan, SUV, truck)
The outcome variable is whether or not the customer ultimately purchases a car from the dealership.

One way that age and income may interact with each other is that older customers with higher incomes may be more likely to purchase expensive cars like SUVs or luxury sedans. To capture this interaction in a decision tree classifier, we could create a new feature called "age times income" that represents the product of age and income.

We could then build a decision tree that splits the data based on this new feature. For example, the decision tree might split the data into two groups: one group where "age times income" is greater than a certain value (indicating older customers with higher incomes), and another group where it is less than that value (indicating younger customers with lower incomes).

Within each of these groups, the decision tree could further split the data based on other attributes like gender or car type. For example, within the group of older, high-income customers, the decision tree might split the data based on gender (i.e., male vs. female) to see if there are any gender differences in car preferences.

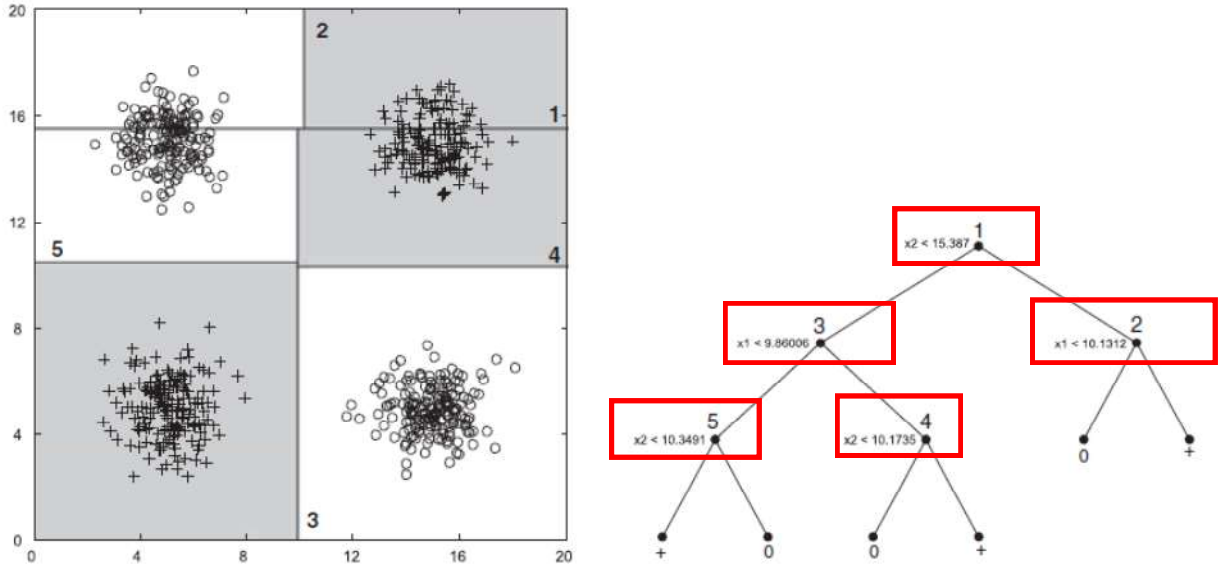
By incorporating interactions between attributes in this way, decision tree classifiers can more accurately model complex relationships between variables and make more accurate predictions about the outcome variable.



In this scatter plot, we have two attributes on the x-axis and y-axis: "Hours Studied" and "Exam Score". The data points represent students' exam scores based on how many hours they studied.

There is clearly a positive correlation between hours studied and exam score. However, there is also an interaction between these attributes - the slope of the line changes at different points along the x-axis. For example, for students who studied less than 4 hours, there is a steep increase in exam score as the hours studied increases. However, for students who studied more than 4 hours, the increase in exam score is more gradual. This suggests that the relationship between hours studied and exam score is not linear and there may be other factors at play that impact exam performance.

Handling interactions



(a) Decision boundary for tree with 6 leaf nodes.

(b) Decision tree with 6 leaf nodes.

Figure 3.28. Decision tree with 6 leaf nodes using X and Y as attributes. Splits have been numbered from 1 to 5 in order of other occurrence in the tree.

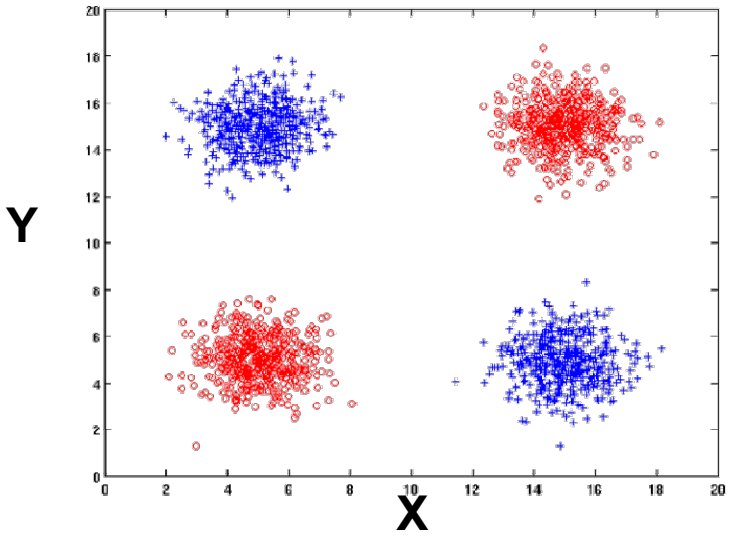
let's consider an example where we are trying to build a decision tree to predict whether or not a customer will purchase a product based on their age and income. Suppose we decide to use a single attribute-based decision boundary at the root node of the tree based on age alone.

If we split the data into two groups based on a threshold age of 30, then all customers younger than 30 would be in one group and all customers older than 30 would be in the other group. Let's suppose that the younger group has a higher proportion of non-purchasers and the older group has a higher proportion of purchasers.

However, if we had used both age and income as features to split the data, we may have found that for customers under 30 with high incomes, they are more likely to purchase the product while for customers over 30 with low incomes, they are less likely to purchase the product.

In this case, using a single attribute-based decision boundary based on age alone would not capture the true relationship between age, income, and purchasing behavior, resulting in a less accurate decision tree model.

Handling interactions given irrelevant attributes



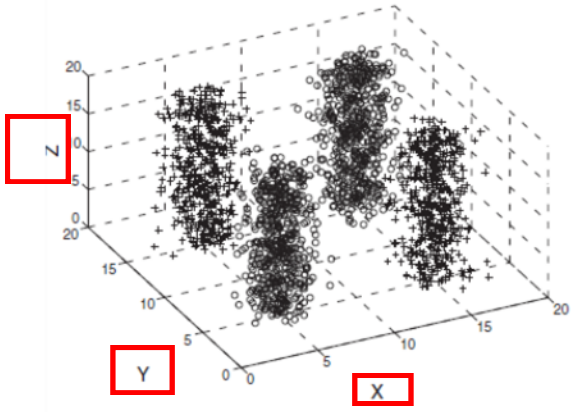
+ : 1000 instances

o : 1000 instances

Adding Z as a noisy attribute generated from a uniform distribution

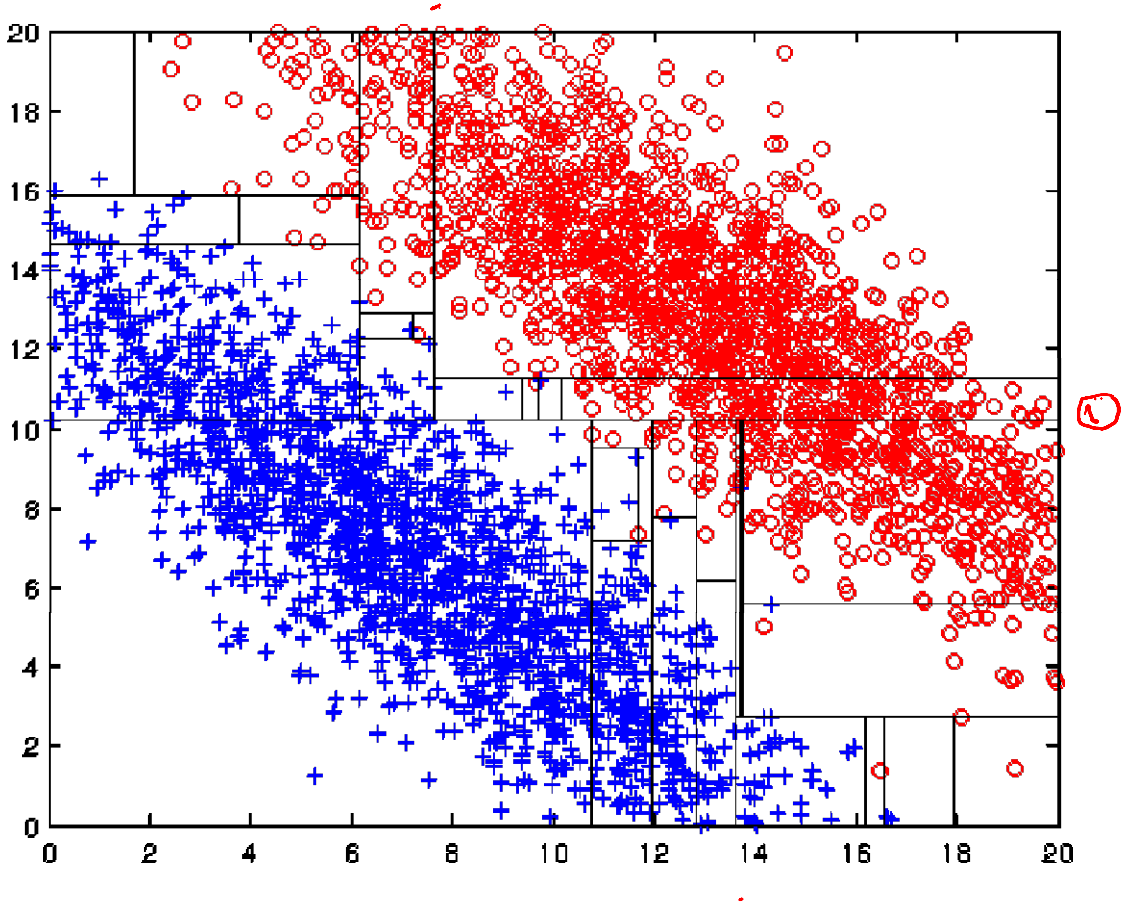
Entropy (X) : 0.99
Entropy (Y) : 0.99
Entropy (Z) : 0.98

Attribute Z will be chosen for splitting!



(a) Three-dimensional data with attributes X, Y, and Z.

Limitations of single attribute-based decision boundaries



Both positive (+) and negative (o) classes generated from skewed Gaussians with centers at (8,8) and (12,12) respectively.

دوتا کلاس ابی و قرمز داریم که کلاس ابی ها ایکس و وای هاشون کم باشه
کلاس قرمزها ایکس و وای هاشون زیاد باشه اگه این مقادیر را به درخت بدیم چیکار میکنه؟ باید دنبال حد استانه ای بگرده که این اده ها را بیشتر از هم متمایز کنه بعد که حد آستانه مشخص شد داده هارو در شاخه های درخت پخش میکنه

مشکل درخت تو این شکل چیه؟
باید بیست بار یا سی بار از دوتا اتریبیوتی که داریم با حداستانه های مختلف استفاده کنه تا بتوانه شبکه بندی و گروه بندی کنه داده هارا
نحوه ی نگاه کردنش و شبکه بندی کردنش به این شکل به درد این مساله نمیخوره