Data Mining

Lecture 3: Classification

Lecturer: G. Gray

Lecture Overview

1. What is classification?
2. Training a classifier
   - Training and test dataset
   - Cross Validation, holdout and bootstrap sampling
3. Evaluating a classifier
   1. Accuracy
   2. Precision
   3. Recall

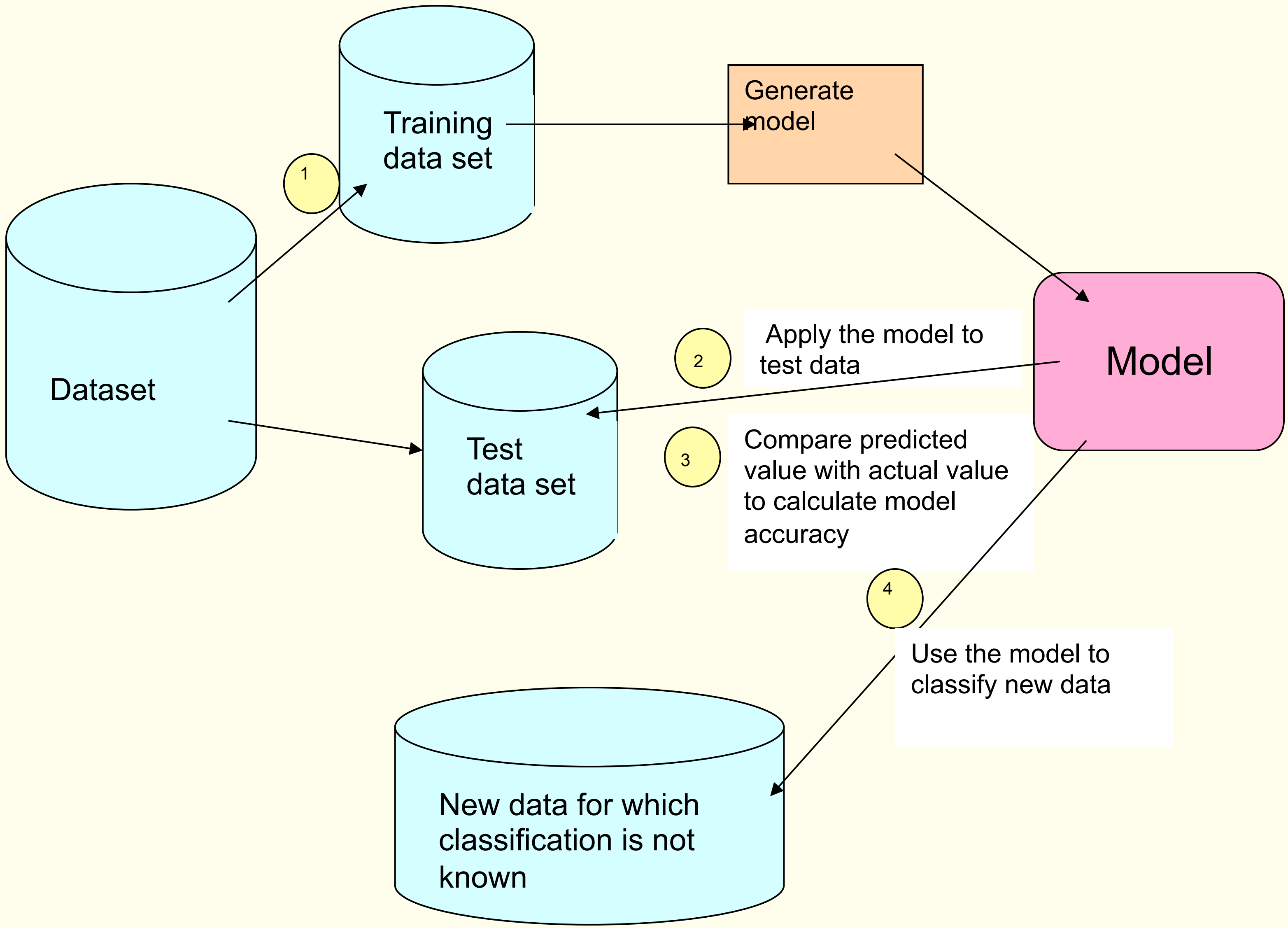
Classification

Predicts a **class label** (the attribute you want to predict) that has a small number of discrete values, for example:

- will customers churn or not (two classes)
- Iris identification (three classes)
- credit approval: identifying if a credit/loan application is a good or bad risk (two classes)

Classification: classifies data (**constructs a model**) based on a training data set and then uses this model to classify new data

Classification Process



Classification Process

Starting with a dataset that has one attributed declared as the class label:

1. Using a portion of the dataset (training dataset), train a model to distinguish between each class in the dataset
2. Use the rest of the dataset to test how accurate the model is (test dataset), i.e. if the model is applied to data that was not available during training, who well does it do?
3. Repeat steps 1 and 2 until you are happy with the model accuracy. Each iteration can try either a different model, or different preprocessing steps.
4. Once you are happy with the model, it is then ready to be applied to data for which the class label is not known, e.g new customers. This is often called **scoring**.

Training a model:

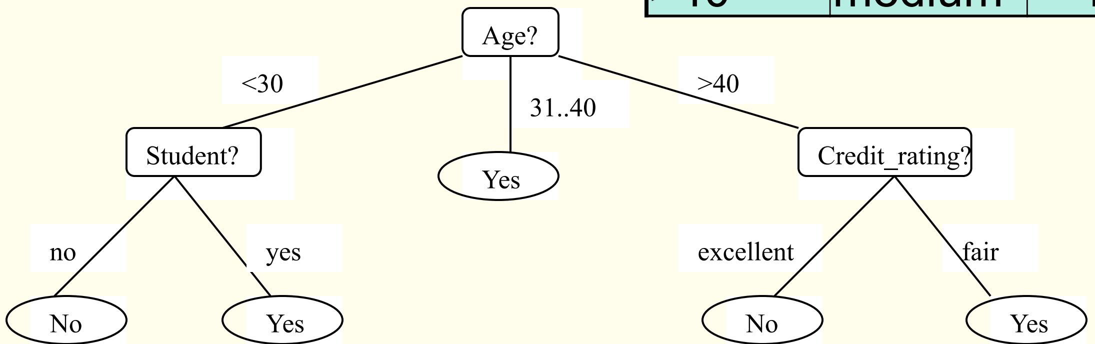
Example of training a decision tree model:

What is a Decision Tree?

- A tree structure where each internal node denotes a test on an attribute.
- Each branch represents an outcome of the test,
- and each leaf node represents a class, or class distribution.

Training dataset

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no



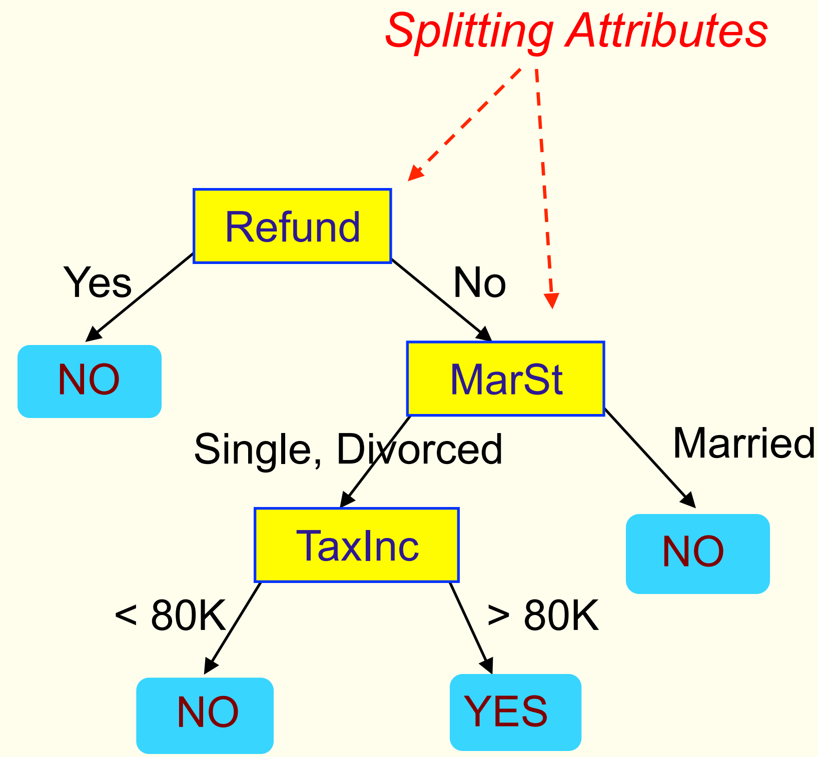
A decision tree indicating if a customer is likely to buy a computer (from Han, Kamber)

Another Example of a Decision Tree

categorical categorical continuous Class label

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Training Data

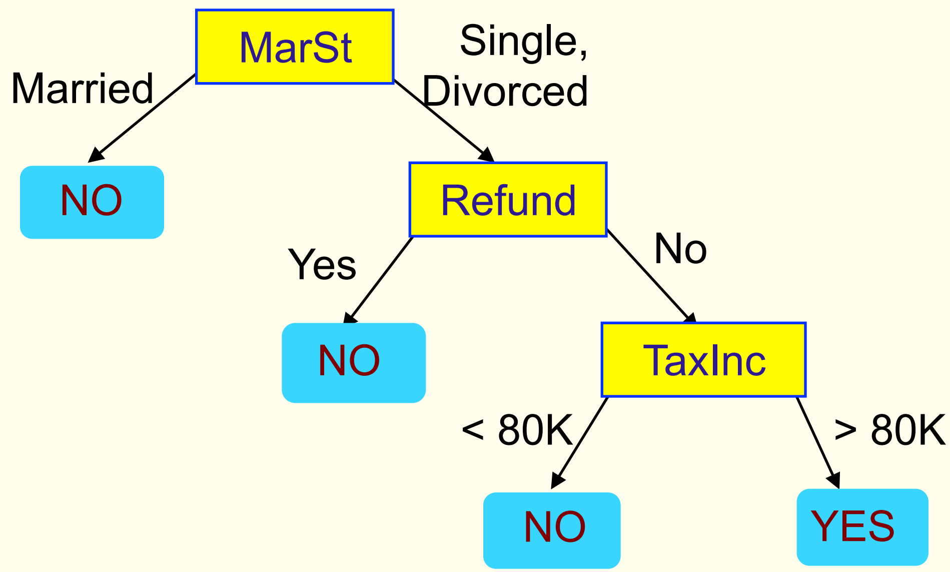


Model: Decision Tree

Another Example of Decision Tree

categorical categorical continuous Class label

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	**No**
2	No	Married	100K	**No**
3	No	Single	70K	**No**
4	Yes	Married	120K	**No**
5	No	Divorced	95K	**Yes**
6	No	Married	60K	**No**
7	Yes	Divorced	220K	**No**
8	No	Single	85K	**Yes**
9	No	Married	75K	**No**
10	No	Single	90K	**Yes**



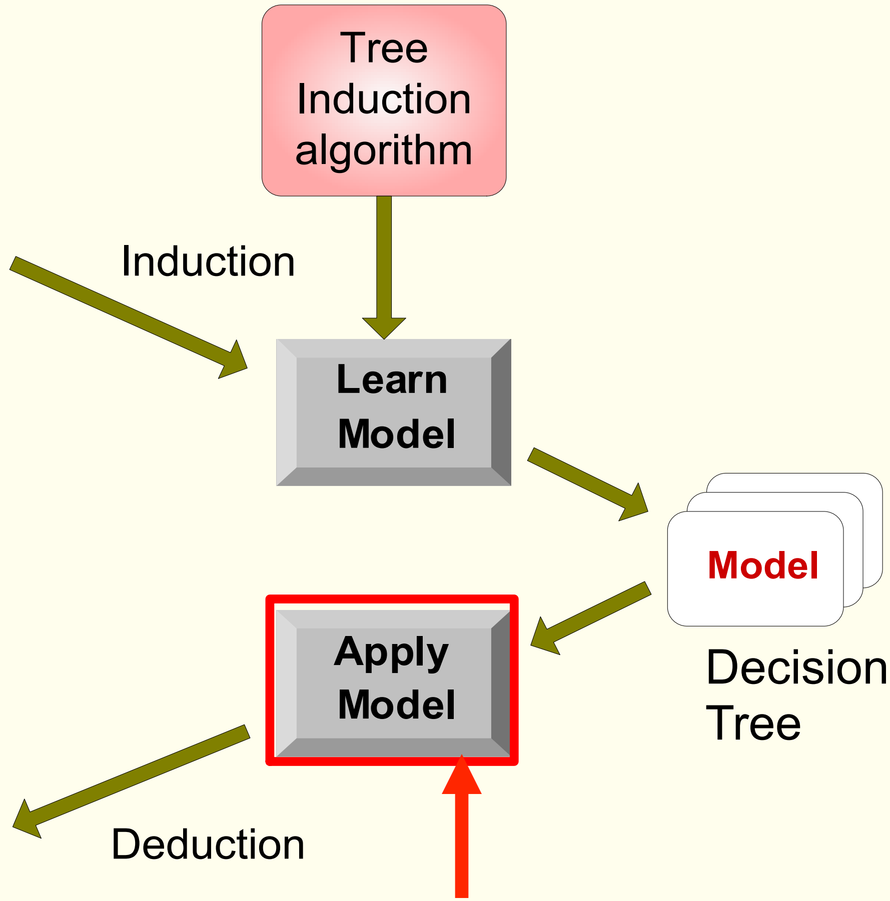
There could be more than one tree that fits the same data!

Using a model:

Using the tree . . .

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

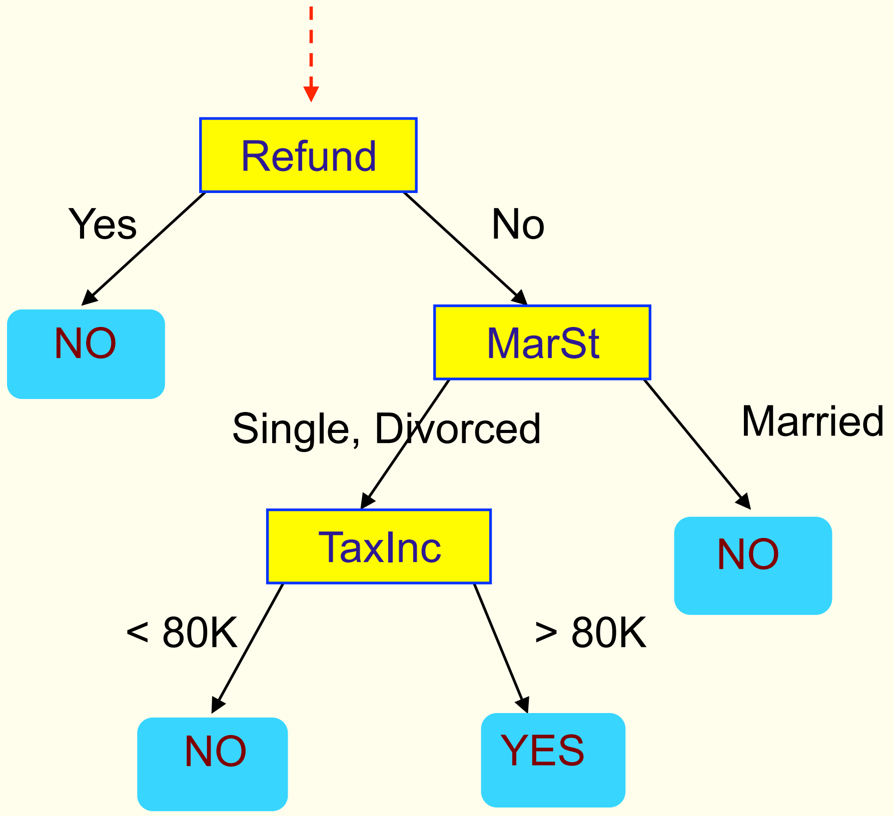
Training Set



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	?

Apply Model to Test Data

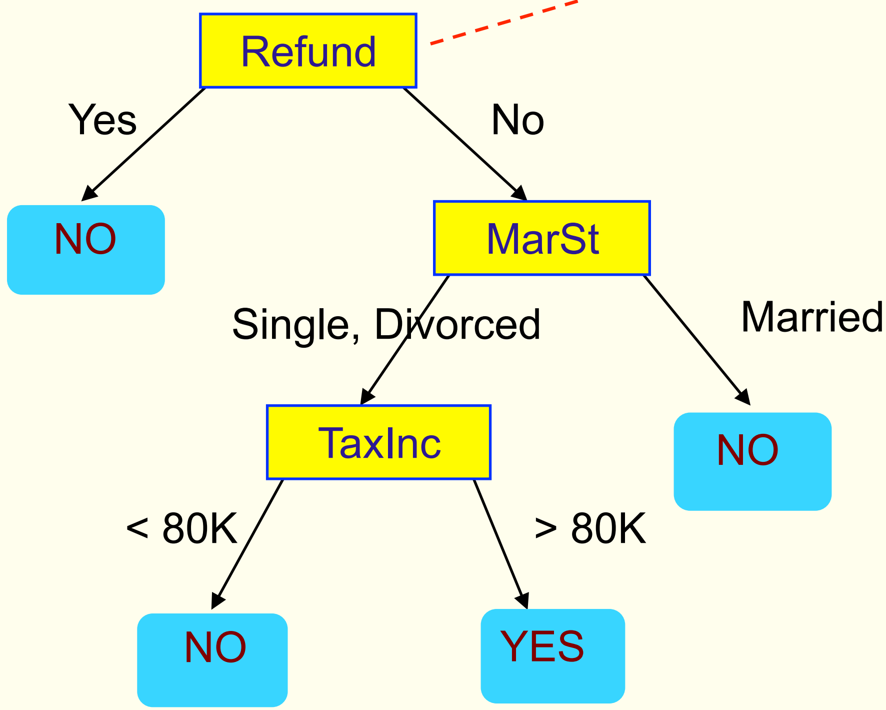
Start from the root of tree.



Test Data

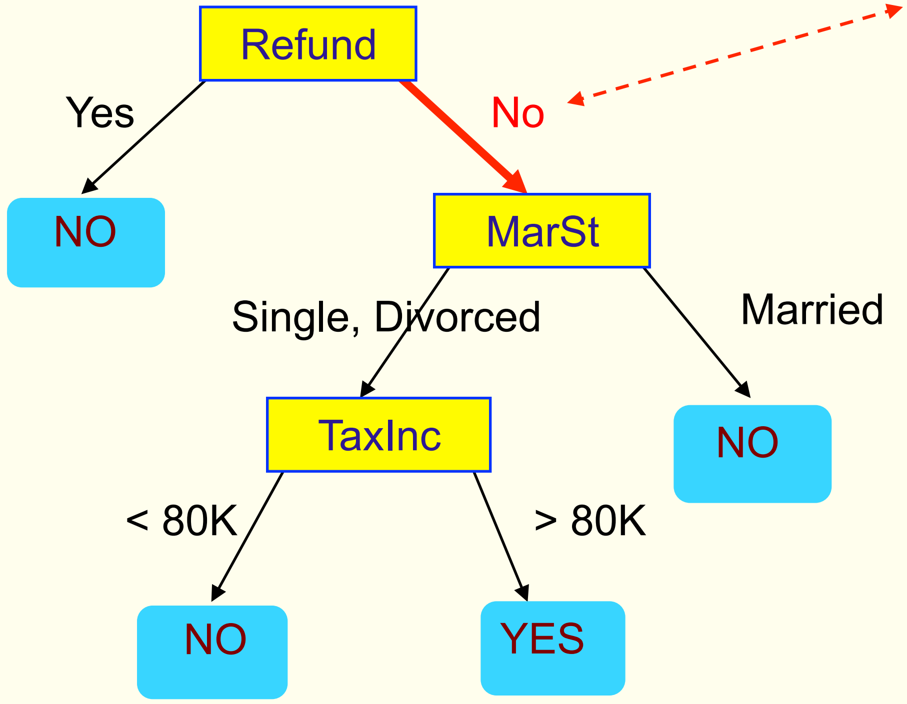
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data



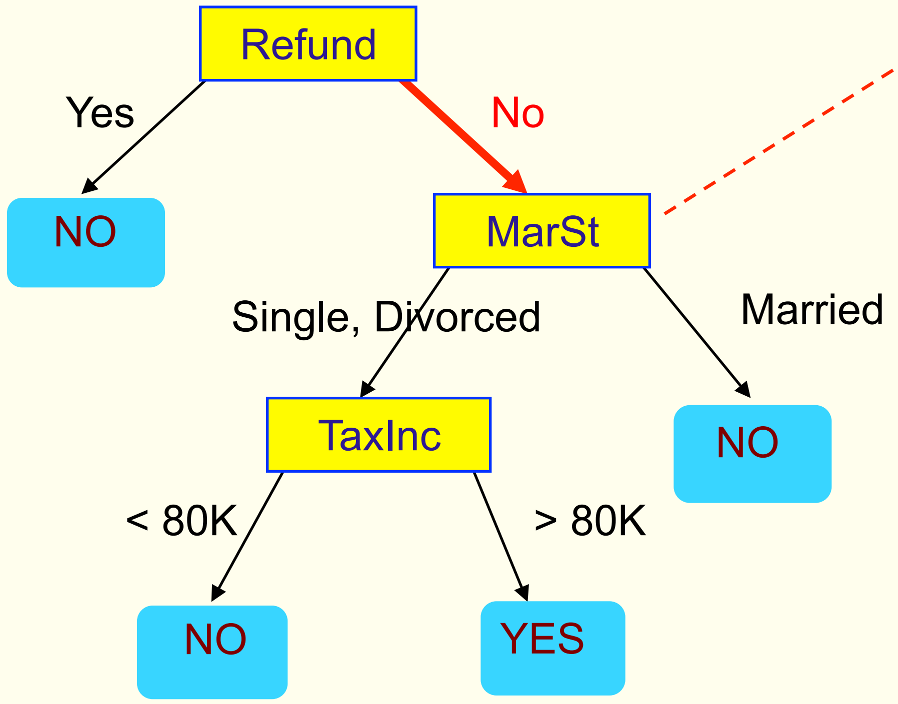
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data



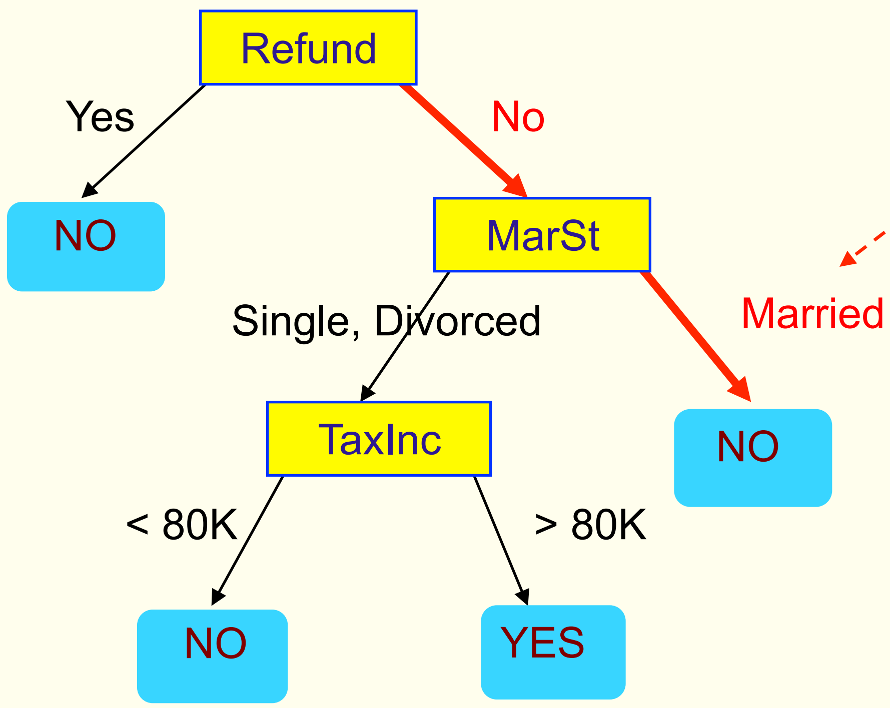
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data



Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

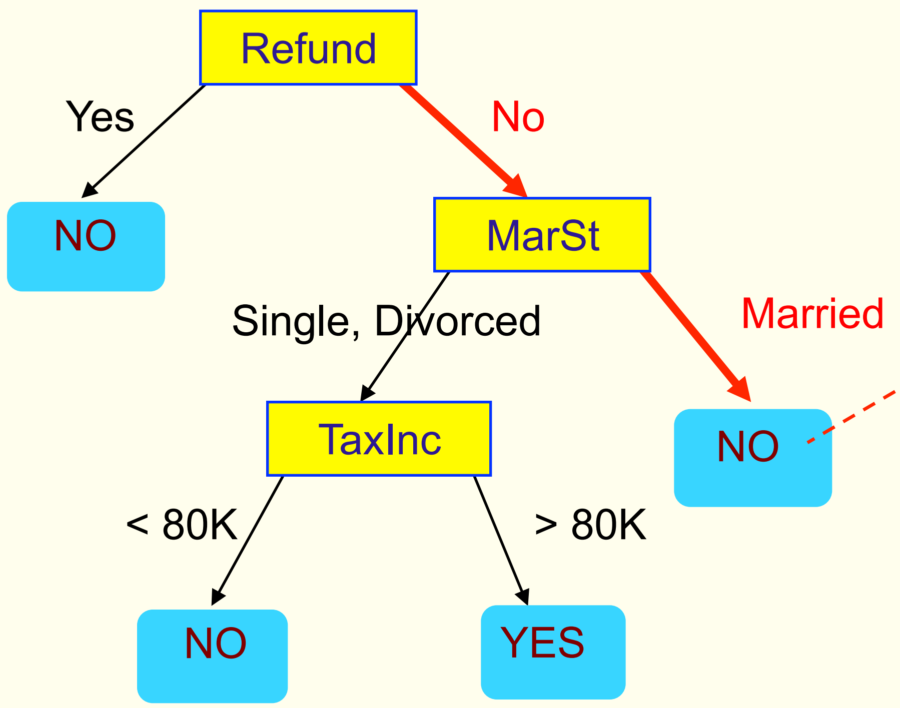
Apply Model to Test Data



Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

Apply Model to Test Data

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



Assign Cheat to “No”

RAPID | MINER

DEMO

Iris & Golf dataset for upcoming slide . . .

How accurate is a model likely to be?

The model knows the patterns in the training dataset . .

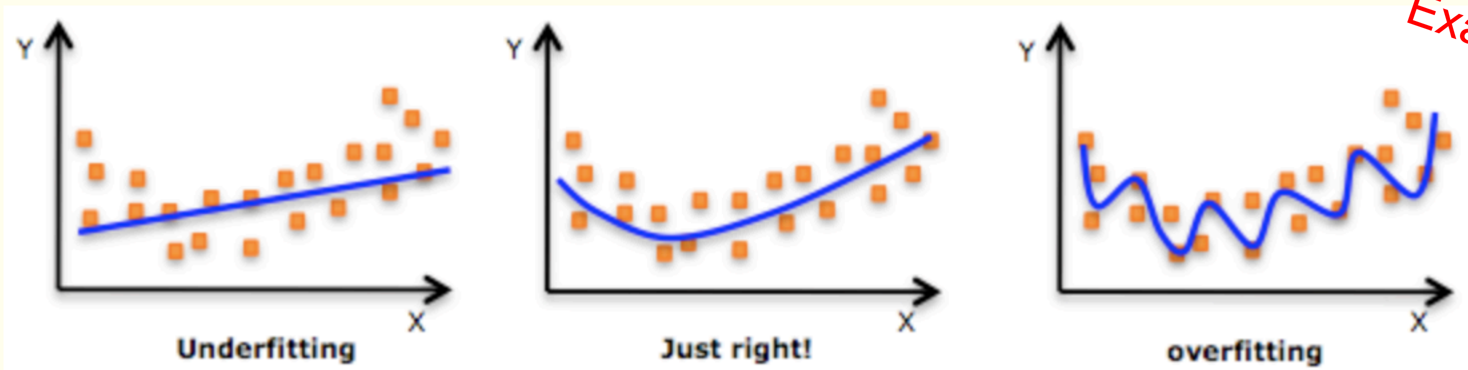
Will the patterns be the same for other data?

Over fitting

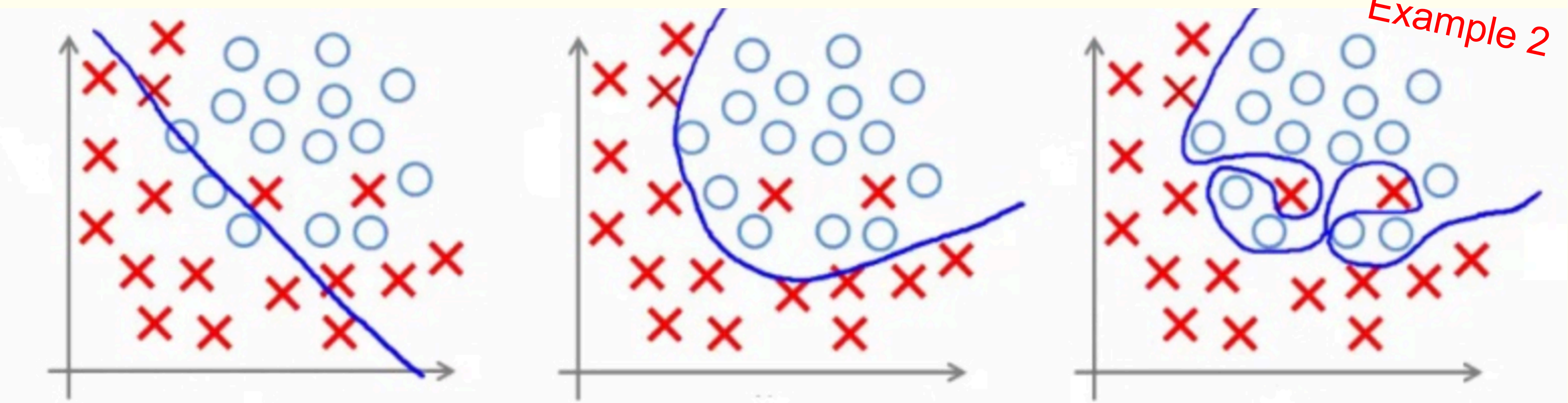
- Classifiers are trained from a training data set and then tested on a test data set.
  - The **training error** is the number of rows misclassified in the training data set.
  - **Generalisation error** is the number of rows misclassified in the testdata set.
- Classifiers are evaluated based on **generalisation error**.
  - This is done to avoid **Model over-fitting**. i.e. when a model becomes to specific (complex) as a result of attempting to accommodate unusual cases in the data.

Model overfitting

Example 1



Example 2



Under-fitting

(too simple to explain the variance)

Appropriate-fitting

Over-fitting

(forcefitting -- too good to be true)

Creating a training and test dataset

- There are a number of ways to split the dataset into a training dataset, and a test dataset, the most common being:

  1. Holdout method

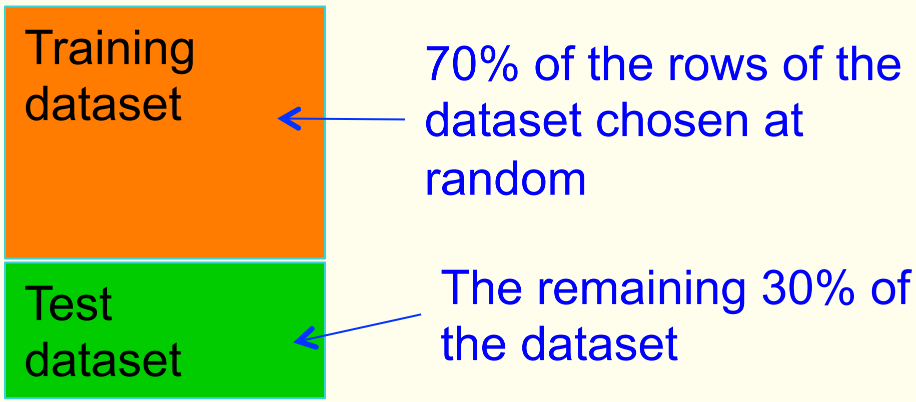
  2. Cross-validation

  And if your dataset is too small . . .

  3. Bootstrap sampling

Holdout method

- The most straight forward method to split the data is the **holdout method**, where the original data set is randomly split in two. One sample is the test data set and the other is the training data set. The original sample is usually split about 70-30, or two thirds for training and one third for testing.



Holdout method

This approach has a number of limitations:

1. **Fewer instances are available** for training because they have been withheld for testing.

2. The **training and test set are not independent** of each other.
   - If a class is over represented in one set, it will be under represented in the other.

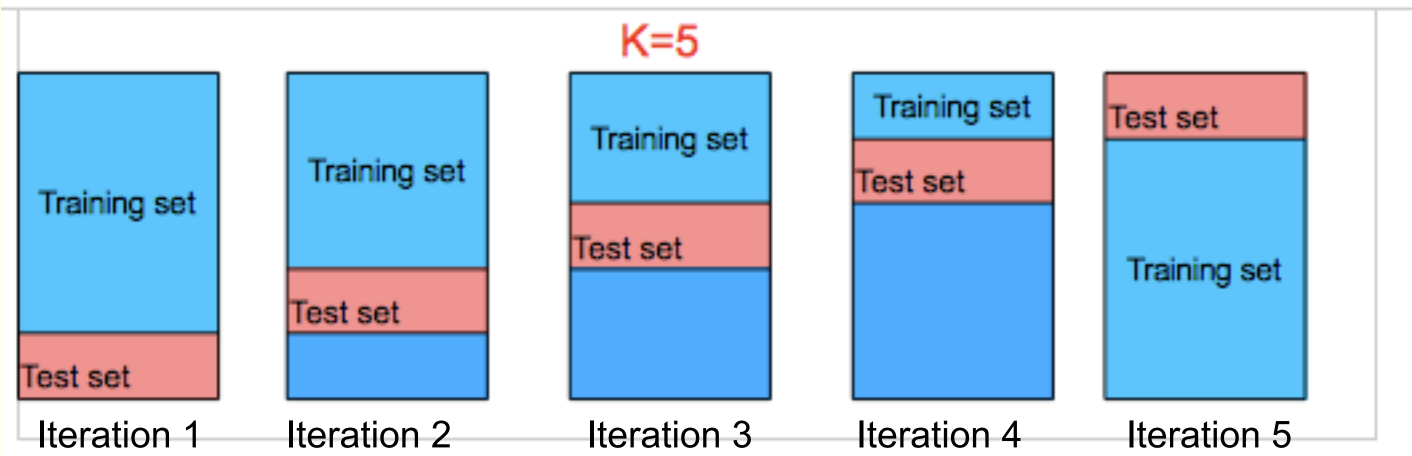
Cross Validation

- By far the most common approach is to use **Cross Validation**:
  - the data set is divided into *k* equal sized partitions.
  - Training is run *k* times. For each run, one sample is used for testing, and the remaining *k-1* samples are used for training.
    - This is done *k* times, ensuring that each partition is used once for testing, and *k-1* times for training.

Remember: this is being done to get a reliable accuracy figure for the model

Cross Validation

- With this approach, the entire data set is used for both training and testing. The value of *k* will determine what percentage of records are used to train the classifier.



- The final model returned is built over the entire dataset, but the accuracy reported is based on the **average** of the errors for each iteration.

Bootstrap sampling

This is not a method to split the data into training & test datasets, but can be done first if the dataset is very small . . .

- **Bootstrapping** is used when the dataset is TOO SMALL to split into a test and training dataset. It is a form of sampling with replacement and works as follows:
- Suppose the original sample has 5 elements: A, B, C, D, E. One element is picked at random and added to the sample. However it is not removed from the original sample and so is available to be picked again. Subsequent elements are picked in the same way. The resulting sample could be something like this:

    A, B, E, D, A, A, D, C, E, B, D etc.

Bootstrap sampling

- Once the dataset is made larger by replicating some rows, it can then be split into a training and test dataset using one of the previous methods.

- The term comes from the expression "Pull yourself up by your bootstraps" meaning you have to use what you've got.
- If sample sizes are too small, bootstrapping, while not ideal, can be used to make a larger dataset with the same patterns.

Rapidminer operators

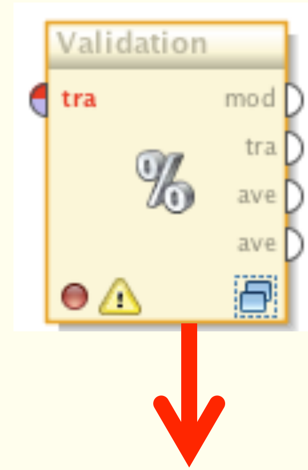
Rapidminer implements all three methods, the operators are found at:

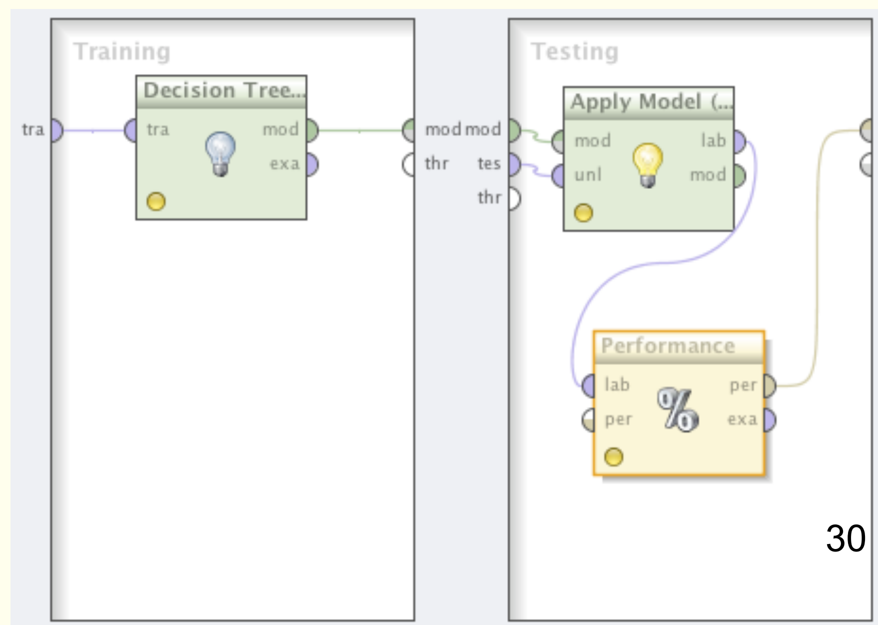
- Holdout: evaluation/validation/split validation
- Cross validation: evaluation/validation/X-validation

- Bootstrap: data transformations/filtering/sampling/sample (bootstrapping)

Rapidminer operators

- Split and X-validation create a parent process which contains two internal sub-processes, one will contain operators for the training dataset, and the other will contain operators for the test dataset.





Evaluating a classifier

Using the training and test dataset

- A classification algorithm will learn the patterns from a labeled training dataset, and produce a model, e.g. a decision tree
- This model will then be applied to the test dataset. The model will give each row in the test dataset a label.
- The performance of the model is evaluated by comparing the <u>actual label</u> of the row with the <u>predicted label</u> the model gave that row.

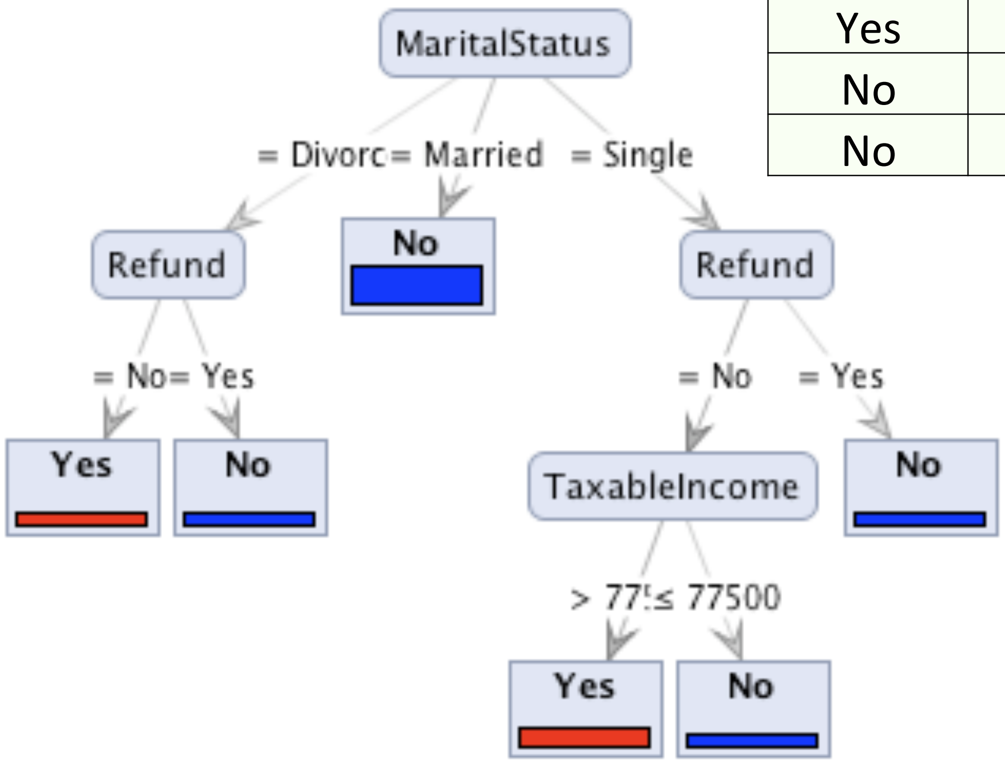
Lets look at an example . . .

- Training dataset:

Refund	MaritalStatus	TaxableIncome	Cheat
Yes	Single	125000	No
No	Married	100000	No
No	Single	70000	No
Yes	Married	120000	No
No	Divorced	95000	Yes
No	Married	60000	No
Yes	Divorced	220000	No
No	Single	85000	Yes
No	Married	75000	No
No	Single	90000	Yes
No	Married	85000	No
Yes	Married	85000	No
Yes	Single	125000	No

Test Dataset

Decision Tree



Refund	MaritalStatus	TaxableIncome	Actual Label Cheat
Yes	Single	125000	No
No	Divorced	125000	No
No	Married	120000	No
No	Single	50000	Yes
Yes	Divorced	120000	No
No	Divorced	80000	Yes
No	Single	80000	No

Actual label.

What label will each row be given by the decision tree?

Which predictions will be correct?

Predicted Value versus Actual Value

			Actual Label	Predicted Label	
Refund	**MaritalStatus**	**TaxableIncome**	**Cheat**	**Predicted Cheat**	
Yes	Single	125000	No	No	✔
No	Divorced	125000	No	Yes	✖
No	Married	120000	No	No	✔
No	Single	50000	Yes	No	✖
Yes	Divorced	120000	No	No	✔
No	Divorced	80000	Yes	Yes	✔
No	Single	80000	No	Yes	✖

✔ Three rows had a label of ‘No’ and were predicted correctly.
✔ One row had a label of ‘Yes’ and was predicted correctly.
✖ One row had a label of ‘Yes’ and was predicted incorrectly.
✖ Two rows had a label of ‘No’ and were predicted incorrectly.

Confusion Matrix

- Typically, the accuracy of a classification model is presented as a confusion matrix.
  - Used to determine if the classification model is confusing the different classes in the dataset.
- The confusion matrix for the example on the last slide is as follows:

		Predicted Class	
		Yes	No
Actual Class	yes	**1**	1
	No	2	**3**

Confusion Matrix

1 was correctly predict as YES

		Predicted Class	
		Yes	No
Actual Class	yes	**1**	1
	No	2	**3**

1 was incorrectly predict as NO

2 were incorrectly predict as YES

3 were correctly predict as NO

Confusion Matrix

		Predicted Class	
		Yes	No
Actual Class	yes	**a**	b
	No	c	**d**

- Other figures than can be calculated from the confusion matrix:
  - The overall accuracy (AC) of the classifier: what percentage of predictions were correct:

$$AC = \frac{a + d}{a + b + c + d}$$

  - Recall: the proportion of cases for a particular class that were predicted correctly. Recall for ‘yes’ above is:

$$Recall = \frac{a}{a + b}$$

  - Precision: the proportion of predictions for a particular class that were correct. Precision for ‘yes’ above is:

$$Precision = \frac{a}{a + c}$$

Confusion Matrix

		Predicted Class	
		Yes	No
Actual Class	yes	**1**	1
	No	2	**3**

- Returning to our previous example:
  - The overall accuracy (AC) of the classifier: what percentage of predictions were correct:

$$AC = \frac{1+3}{1+1+2+3} = \frac{4}{7} = 57\%$$

  - Recall: the proportion of cases for a particular class that were predicted correctly. Recall for 'yes' above is:

$$\text{Recall (yes)} = \frac{1}{1+1} = \frac{1}{2} = 50\%$$

$$\text{Recall (no)} = \frac{3}{2+3} = \frac{3}{5} = 60\%$$

  - Precision: the proportion of predictions for a particular class that were correct. Precision for 'yes' above is:

$$\text{Precision(yes)} = \frac{1}{1+2} = \frac{1}{3} = 33\%$$

$$\text{Precision(no)} = \frac{3}{3+1} = \frac{3}{4} = 75\%$$

Explaining Recall and Precision

- Suppose you enter a query in Google, to which Google responded with 20 documents.
- Suppose we know there were 15 documents on the internet which are relevant to our query . . .
- There are two figures of interest in evaluating how google preformed:
  - How many of the 15 relevant documents did it find? (Recall)
  - Of the twenty docs it did return, how many were relevant (Precision)
- Lets suppose 10 of the 20 documents returned were relevant:
  - Recall = 10 / 15 = 66%
  - Precision = 10 / 20 = 50%

Explaining Recall and Precision

		Predicted Class	
		Yes	No
Actual Class	yes	**a**	b
	No	c	**d**

Similarly with a classifier . . . . .

- Recall is how many actual rows in a class were predicted correctly
  - look across a row, how many actual ‘yes’s’ were predicted as a yes
- Precision is how many of the predicted rows in a class were correct
  - Looking down a column, how many predicted ‘yes’s’ were correct.

Confusion Matrix: more than two classes

- This is an example of a confusion matrix from http://en.wikipedia.org/wiki/Confusion_matrix

Example confusion matrix

		Predicted		
		Cat	Dog	Rabbit
Actual	Cat	5	3	0
	Dog	2	3	1
	Rabbit	0	2	11

- Again the diagonal shows the correct predictions
  - 5 cats predicted correctly
  - 3 dogs predicted correctly
  - 11 Rabbits predicted correctly
- Overall accuracy is 70%

$$AC = \frac{5 + 3 + 11}{5+3+0+2+3+1+0+2+11}$$

$$= 19 / 27 \quad = \quad 70\%$$

- Recall for CAT is 5 / 5+ 3+ 0 = 5/8 = 62.5%

- Precision for CAT is 5 / 5 + 2 + 0 = 5 / 7 = 71%

Classification accuracy

- If all classes are of equal importance, then calculating overall accuracy is enough when evaluating a classifier.
- However, for many applications, some classes are more important to get right than others, because the cost of an error is different for different classes.
  - For example, the error committed in diagnosing someone as healthy when one has a life-threatening illness (known as a false negative decision) is far more serious than diagnosing someone as ill when one is in fact healthy (known as a false positive).
- A *confusion matrix* gives a breakdown of the accuracy for each class.

Exercise

		Predicted Class		
Actual Class		setosa	veriscolor	virginica
	setosa	**50**	0	0
	veriscolor	0	**43**	7
	virginica	0	5	**45**

- Given the confusion matrix above for 150 examples from the iris dataset, how would you calculate:
  - The overall accuracy of the classifier:
  - The recall and precision for iris-setosa
  - The recall and precision for iris-versicolor

Confusion Matrix in RapidMiner

- RapidMiners ‘Performance (Classification)’ operator generates a confusion matrix from applying a model to a test dataset.
- The following is the output from the sample rapidminer process found under repositories at: samples/processes/03_Validation/06_ConfusionMatrix
- The dataset has 500 rows, and four classes.

accuracy: 61.20% +/- 6.14% (mikro: 61.20%)

	true one	true two	true three	true four	class precision
pred. one	156	23	19	51	62.65%
pred. two	32	55	0	0	63.22%
pred. three	20	0	24	0	54.55%
pred. four	48	1	0	71	59.17%
class recall	60.94%	69.62%	55.81%	58.20%	

Note: Prediction figures are across the rows, and actual figures are down the columns, unlike the typical layout for a confusion matrix:

Is precision or recall more important?

Take a dataset where the class label is: medical condition –Yes / No. 5 people have the condition, and 20 people do not.

If a model is very conservative regarding making a diagnosis, i.e unlikely to make a diagnosis unless absolutely sure, then precision will be high, but recall will be low. If it predicts you have the condition, it will be correct, but it may miss people who do have the condition.

		Predicted Class		
Actual Class		Yes	No	recall
	yes	**1**	4	20%
	No	0	**20**	
precision		100%		

		Predicted Class		
Actual Class		Yes	No	recall
	yes	**5**	0	100%
	No	20	**0**	
precision		20%		

If a model is not conservative about making a diagnosis, i.e. will give a diagnosis even if there is a small chance you have the condition, then precision will be low, but recall will be high. It will catch everyone who has the condition, and some additional people as well.

A good model will try to maximise recall and precision

Which of the following predictions would you trust?

1. A model says the customer you are talking to will churn
   - The model has high precision for churn=yes

Yes/No

2. A model says the customer you are talking to will churn
   - The model has high recall for churn=yes

Yes/No

3. A model says the customer you are talking to will churn
   - The model has high recall for churn=yes, but low precision for churn=yes.

Yes/No

4. 4. A model returns a list of customers that will churn.
   - The model has high precision but low recall for churn=yes

Yes/No

Summary

- Modelling data:
  - Classification
    - Train a model on the training dataset
    - Test for accuracy on a test dataset
    - Apply the model to unlabelled data
  - Overfitting and Under-fitting
    - Models are assesed on **generalisation error** to avoid under and over fitting
  - Confusion matrix
    - Overall model accuracy
    - Class precision
    - Class recall