Data Mining

Classification

- Part 2 -

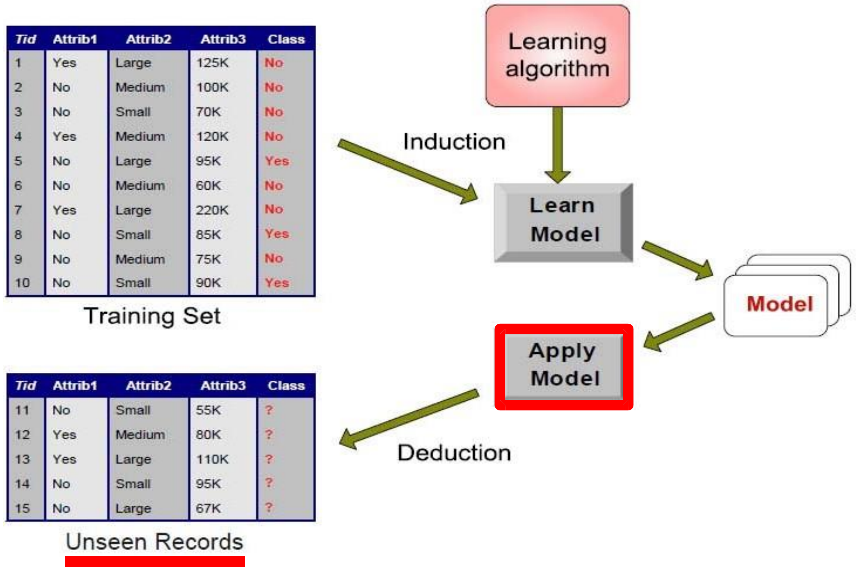
Outline

4. Model Evaluation

Central Question:

How good is a model at classifying unseen records?

(generalization performance)



1. Metrics for Model Evaluation
   - How to measure the performance of a model?

2. Methods for Model Evaluation
   - How to obtain reliable estimates?

4.1 Metrics for Model Evaluation

– Focus on the predictive capability of a model
  - rather than how much time it takes to classify records or build models
– The confusion matrix counts the correct and false classifications
  - the counts are the basis for calculating different performance metrics

Confusion Matrix

	PREDICTED CLASS		
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	True Positives	False Negatives
	Class=No	False Positives	True Negatives

Accuracy and Error Rate

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{\text{Correct predictions}}{\text{All predictions}}$$

$$\text{Error Rate} = 1 - Accuracy$$

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	TP 25	FN 4
	Class=No	FP 6	TN 15

$$\text{Acc} = \frac{25 + 15}{25 + 15 + 6 + 4} = 0.80$$

The Class Imbalance Problem

- Sometimes, classes have very unequal frequency
  - Fraud detection: 98% transactions OK, 2% fraud
  - E-commerce: 99% surfers don’t buy, 1% buy
  - Intruder detection: 99.99% of the users are no intruders
  - Security: >99.99% of Americans are not terrorists
- The class of interest is commonly called the positive class and the rest negative classes
- Consider a 2-class problem
  - number of negative examples = 9990
    number of positive examples = 10
  - if model predicts all examples to belong to the negative class, the accuracy is 9990/10000 = 99.9 %
  - Accuracy is misleading because model does not detect any positive example

Precision and Recall

Alternative: Use performance metrics from information retrieval which are biased towards the positive class by ignoring TN

Precision *p* is the number of correctly classified positive examples divided by the total number of examples that are classified as positive

Recall *r* is the number of correctly classified positive examples divided by the total number of actual positive examples in the test set

$$p = \frac{TP}{TP + FP} \qquad r = \frac{TP}{TP + FN}$$

	Classified Positive	Classified Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

Ignored majority

Precision and Recall – A Problematic Case

	Classified Positive	Classified Negative
Actual Positive	1	99
Actual Negative	0	1000

– This confusion matrix gives us

precision p = 100%

recall r = 1%

– because we only classified one positive example correctly and no negative examples wrongly

– Thus, we want a measure that

1. combines precision and recall and
2. is large if both values are large

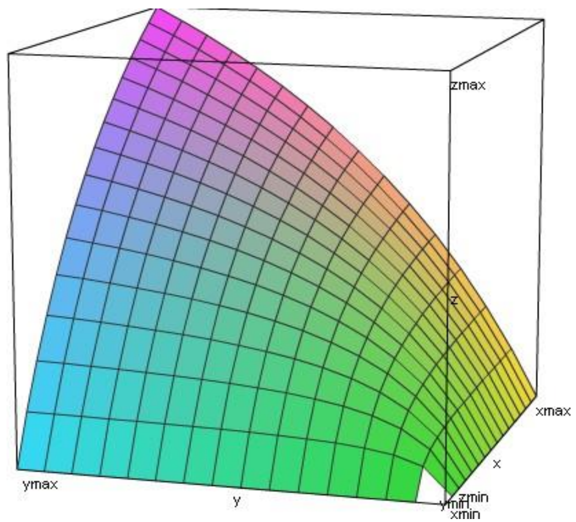
F_1-Measure

- F_1-score combines precision and recall into one measure
- F_1-score is the harmonic mean of precision and recall
  - the harmonic mean of two numbers tends to be closer to the smaller of the two
  - thus for the F_1-score to be large, both *p* and *r* must be large

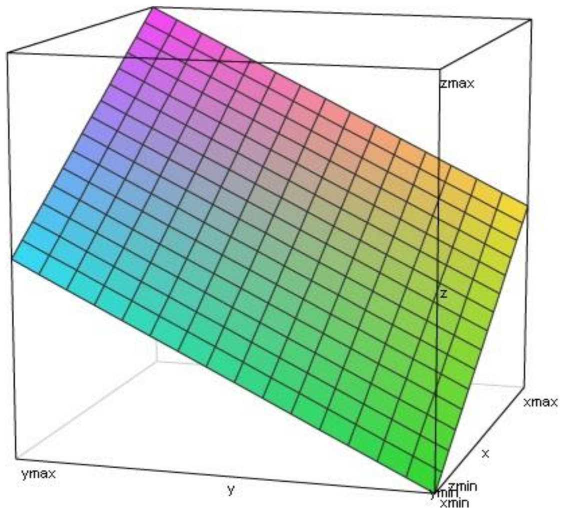
$$F_1 = \frac{2pr}{p+r}$$

$$= \frac{2TP}{2TP + FP + FN}$$

Harmonic mean



Arithmetic mean



Example: Alternative Metrics on Imbalanced Data

	PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	10	0
	Class=No	10	980

$$\text{Precision (p)} = \frac{10}{10+10} = 0.5$$

$$\text{Recall (r)} = \frac{10}{10+0} = 1$$

$$F_1\text{ - measure }(F_1) = \frac{2*1*0.5}{1+0.5} = 0.62$$

$$\text{Accuracy} = \frac{990}{1000} = 0.99$$

	PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	1	9
	Class=No	0	990

$$\text{Precision (p)} = \frac{1}{1+0} = 1$$

$$\text{Recall (r)} = \frac{1}{1+9} = 0.1$$

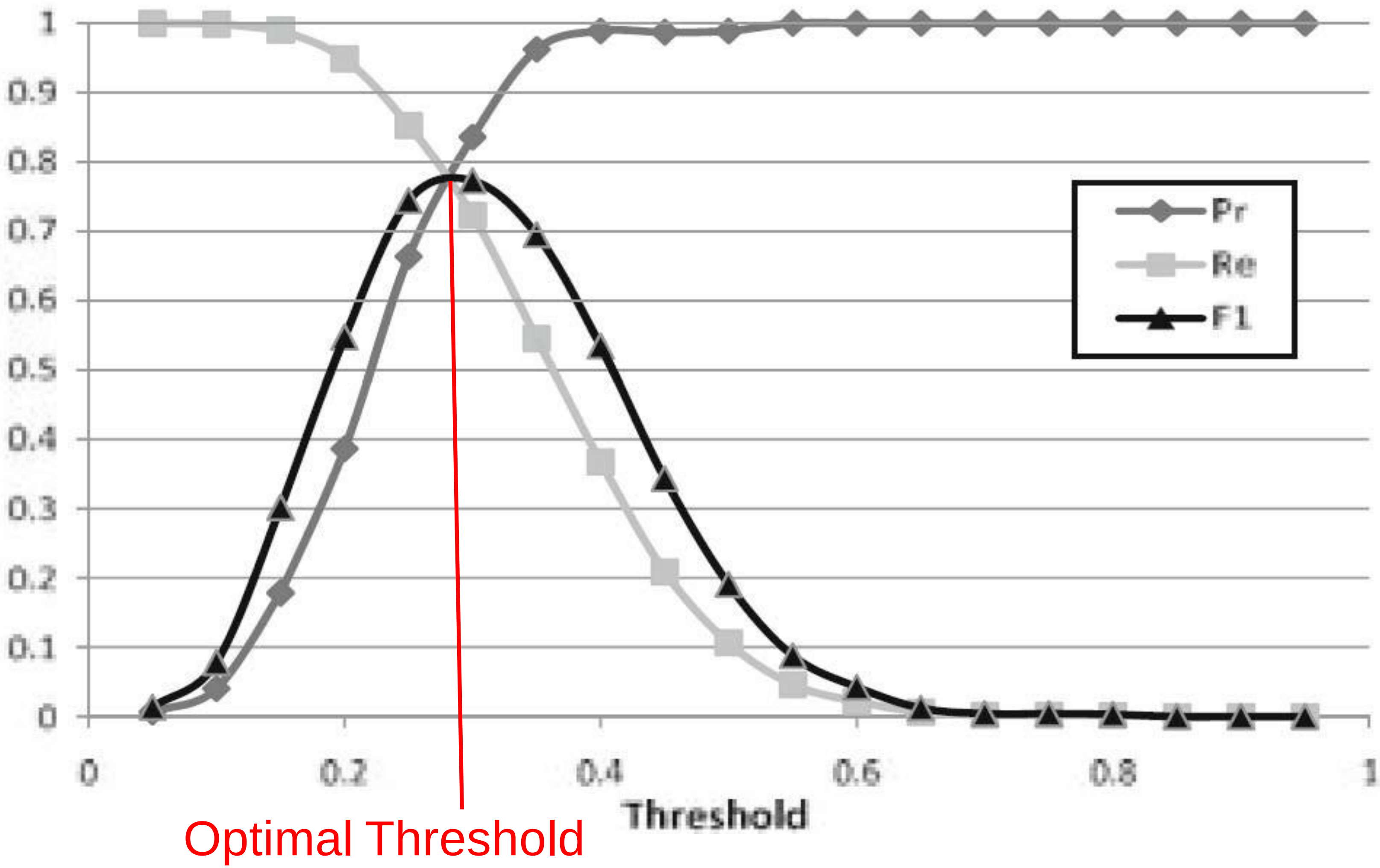
$$F_1\text{ - measure }(F_1) = \frac{2*0.1*1}{1+0.1} = 0.18$$

$$\text{Accuracy} = \frac{991}{1000} = 0.991$$

F_1-Measure Graph

Low threshold: Low precision, high recall

Restrictive threshold: High precision, low recall

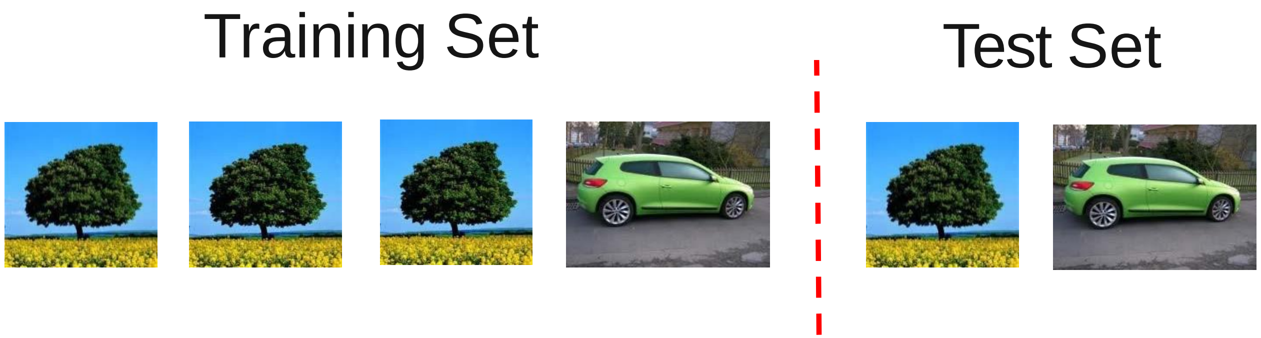


4.2 Methods for Model Evaluation

- How to obtain a reliable estimate of the generalization performance?
- General approach: Split set of labeled records into a training set and a test set
- Never ever test a model on data that was used for training!
  - Because model has been fit to training data, evaluating on training data does not result in a suitable estimate of the performance on unseen data
  - We need to keep training set and test set strictly separate
- Which labeled records to use for training and which for testing?
- Alternative splitting approaches:
  1. Holdout Method
  2. Random Subsampling
  3. Cross Validation

Holdout Method

- The holdout method reserves a certain amount of the labeled data for testing and uses the remainder for training
- Usually: 1/3 for testing, 2/3 for training (or even better 20% / 80%)

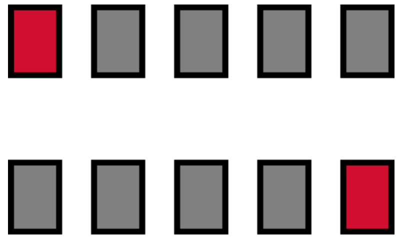


Random Subsampling

– Holdout estimate can be made more reliable by repeating the process with different subsamples
  - in each iteration, a certain proportion is randomly selected for training
  - the performance of the different iterations is averaged

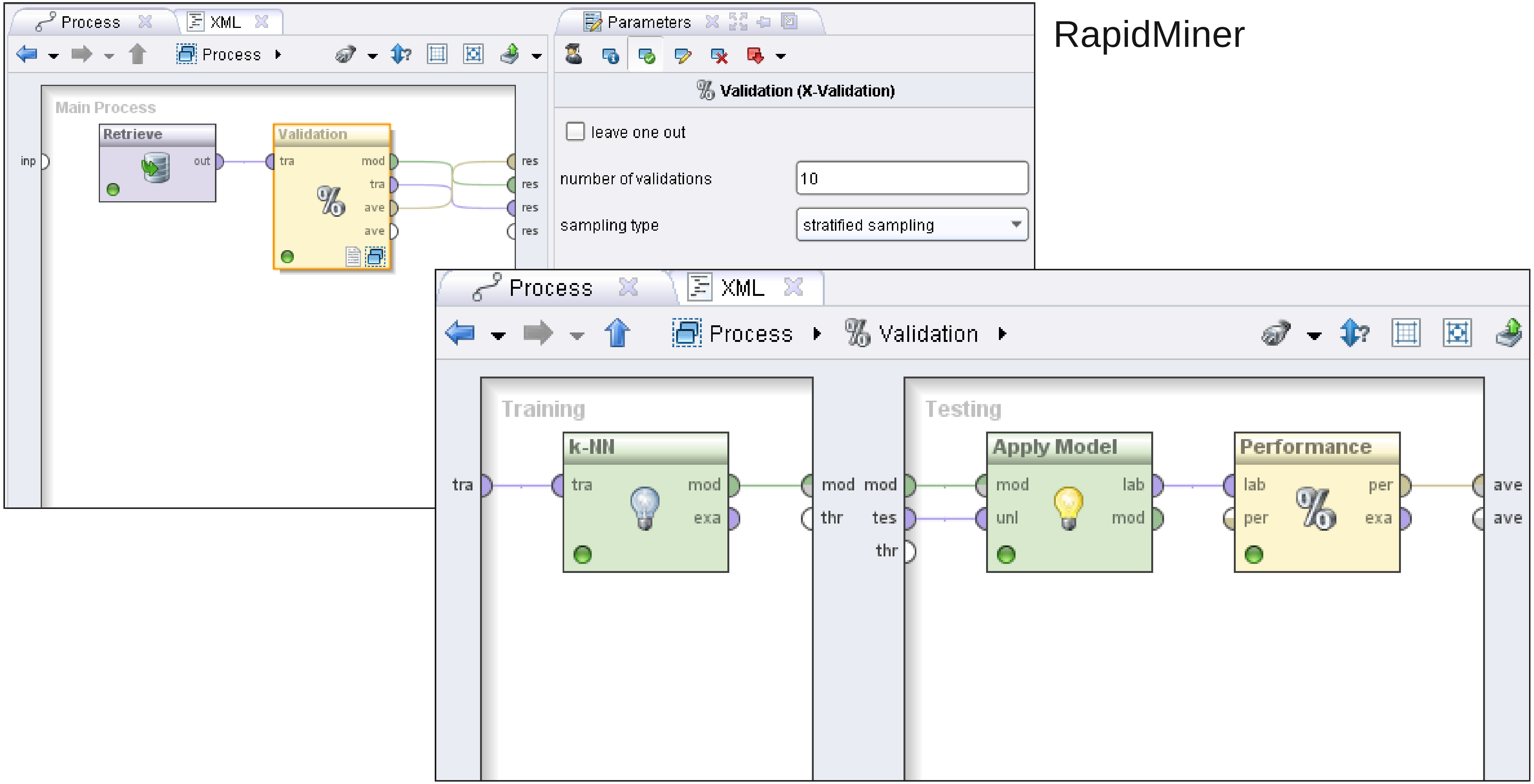
– Still not optimal as the different test sets may overlap
  1. problem: some outliers might always end up in the test sets
  2. problem: important records for learning (red tree) might always be in test sets

Cross-Validation



- Cross-validation avoids overlapping test sets
  - first step: data is split into *k* subsets of equal size
  - second step: each subset in turn is used for testing and the remainder for training
  - this is called k-fold x-validation
- Every record is used exactly once for testing
- The performance estimates of all runs are averaged to yield overall performance estimate
- Frequently used: k = 10 (90% training, 10% testing)
  - why ten? Experiments have shown that this is the good choice to get an accurate estimate and still use as much data as possible for training

Cross-Validation in RapidMiner



RapidMiner

Cross-Validation Results in RapidMiner

Average accuracy over all 10 runs (test sets)

Standard deviation of accuracy values over all 10 runs (test sets)

accuracy: 92.00% +/- 5.26% (micro average: 92.00%)

	true Iris-setosa	true Iris-versicolor	true Iris-virginica	class precision
pred. Iris-setosa	50	0	0	100.00%
pred. Iris-versicolor	0	46	8	85.19%
pred. Iris-virginica	0	4	42	91.30%
class recall	100.00%	92.00%	84.00%	

Recall given that we define Iris-setosa as positive class

Number of correctly classified Iris-versicolar examples in all runs (test sets)

Each record is used exactly once for testing ➔ The numbers in the confusion matrix sum up to the size of the labeled dataset

Evaluation Summary

- Performance metrics
  - Default: Use accuracy
  - If interesting class is infrequent, use precision, recall, and F1
- Estimation of metric
  - Default: Use cross-validation
  - If labeled dataset is large (>5000 examples) and
    - computation takes too much time or
    - exact replicability of results matters, e.g. for data science competitions

  use the holdout method with fixed split