# COMP30027 Machine Learning
# Evaluation, part I

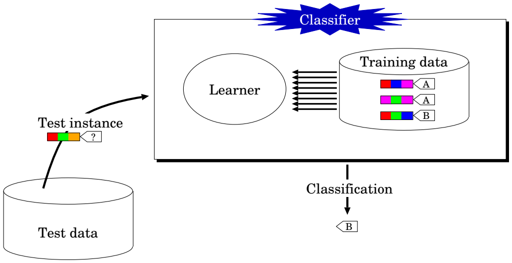
Semester 1, 2019

Jeremy Nicholson & Tim Baldwin & Karin Verspoor

## Lecture Outline

## The Nature of "Classification"

- Input: set of labelled training instances; set of unlabelled test instances
- Model: an estimate of the underlying target function
- Output: prediction of the classes of the test instances



## What is a good Classifier? I

Our goal (in a supervised Machine Learning framework):

- Have a perfect model?
  - Not necessarily clear what that means
  - More difficult than necessary
- Make predictions that are correct!

## What is a good Classifier? II

The basic **evaluation metric**: Accuracy

$$\text{Accuracy} = \frac{\text{Number of correctly labelled test instances}}{\text{Total number of test instances}}$$

Quantifies how frequently the classifier is correct, with respect to a fixed dataset with known labels
(Other metrics attempt to evaluate different underlying behaviour of the target function)

## Evaluating Classification I

Main idea:

- Train (build classifier) using **training data**
- Test (evaluate classifier) on **test data**
    - For instances in the test data, compare predicted class label with actual class label

But often, we just have **data** — a collection of instances

## Evaluating Classification II

Recommended strategy for Project 1, but **strongly not recommended** in ML more generally:

- Use all of the instances as training data
    - Build the classifier using all of the instances
- Use all of the (same) instances as test data
    - Evaluate the classifier using all of the instances

"Testing on the training data" tends to grossly over-estimate classifier performance.
Effectively, we are telling the classifier what the correct answers are, and then asking whether it can come up with the correct answers.

## Holdout I

One solution: **Holdout** evaluation strategy

- Each instance is randomly assigned as either a training instance **or** a testing instance
- Effectively, the data is **partitioned** — no overlap between datasets
- Evaluation strategy:
    - Build the classifier using (only) the training instances
    - Evaluate the classifier using (only) the (different) test instances

Very commonly used strategy; typical split sizes are approximately 50–50, 80–20, 90–10 (train, test)

**Source(s):** Tan et al. [2006, pp 186–7]

## Holdout II

- Advantages:
  - simple to work with and implement
  - fairly high reproducibility
- Disadvantages:
  - size of the split affects estimate of the classifier's behaviour:
    - lots of test instances, few training instances: learner doesn't have enough information to build an accurate model
    - lots of training instances, few test instances: learner builds an accurate model, but test data might not be representative (so estimates of performance can be too high/too low)

## Repeated Random Subsampling I

Slower, but somewhat better solution: **Repeated Random Subsampling**

- Like Holdout, but iterated multiple times:
    - A new training set and test set are randomly chosen each time
    - Relative size of training–test is fixed across iterations
    - New model is built each iteration
- Evaluate by averaging (chosen metric) across the iterations

## Repeated Random Subsampling II

- Advantages:
    - averaging Holdout method tends to produce more reliable results
- Disadvantages:
    - more difficult to reproduce
    - slower than Holdout (by a constant factor)
    - wrong choice of training set–test set size can still lead to highly misleading results (that are now very difficult to sanity–check)

## Cross–Validation I

Usually preferred alternative: **Cross–Validation**

- Data is progressively split into a number of partitions $m$ ($\geq 2$)
- Iteratively:
    - One partition is used as test data
    - The other $m-1$ partitions are used as training data
- Evaluation metric is aggregated across $m$ test partitions
    - This could mean averaging, but more often, counts are added together across iterations

## Cross–Validation II

Why is this better than Holdout/Repeated Random Subsampling?

- **Every** instance is a test instance, for some partition
    - Similar to testing on the training data, but without dataset overlap
    - Evaluation metrics are calculated with respect to a dataset that looks like the entire dataset (i.e. the entire dataset)
- Takes roughly the same amount of time as Repeated Random Subsampling (but see below)
- Very reproducible
- Can be shown to minimise **bias** and **variance** of our estimates of the classifier's performance (more on this in Evaluation II)

## Cross–Validation III

How big is $m$?

- Number of folds directly impacts runtime *and* size of datasets:
  - Fewer folds: more instances per partition, more variance in performance estimates
  - More folds: fewer instances per partition, less variance but slower
- Most common choice of $m$: 10 (occasionally, 5)
  - Mimics 90–10 Holdout, but far more reliable
- Best choice: $m{=}N$, the number of instances (known as **Leave–One–Out Cross–Validation**):
  - Maximises training data for the model
  - Mimics actual testing behaviour (every test instance is treated as an individual test "set")
  - Far too slow to use in practice

## Stratification I

- **Inductive Learning Hypothesis:** Any hypothesis found to approximate the target function well over (a sufficiently large) training data set will also approximate the target function well over **unseen** test examples.
- But, Machine Learners suffer from **inductive bias** — assumptions must be made about the data to build a model and make predictions
    - Different assumptions will lead to different predictions
    - Can only sensibly criticise assumptions with respect to **actual data** (i.e. this is an **empirical problem**)
- The only way to guarantee optimal performance over an unseen test set is to know *a priori* what the unseen data set will look like: **No free lunch in supervised learning**!

**Source(s):** Wolpert and Macready [1997]

## Stratification II

Typical inductive bias in our evaluation framework (n.b. independent of model): **Stratification**

- Assume that **class distribution** of unseen instances will be the same as distribution of seen instances
    - Class distribution is used here to extend definitions from continuous domain (regression) to discrete domain (class'n)
    - We'll see this again in Evaluation II
- When constructing Holdout/Cross–Validation partitions, ensure that training data and test data **both** have same class distribution as dataset as a whole
    - Also (occasionally) called "vertical sampling"
    - Analogous with geological stratum (visualising using stacked bars, for example)

## What is a good Classifier? I

The basic **evaluation metric**: Accuracy

$$\text{Accuracy} = \frac{\text{Number of correctly labelled test instances}}{\text{Total number of test instances}}$$

Quantifies how frequently the classifier is correct
Train on a fixed training dataset, test on a fixed testing dataset
— can compare classifiers by using the same dataset partition

## What is a good Classifier? II

| Outlook | Temperature | Humidity | Windy | Actual | Classified |
|---|---|---|---|---|---|
| sunny | hot | high | FALSE | no | |
| sunny | hot | high | TRUE | no | |
| overcast | hot | high | FALSE | yes | |
| rainy | mild | high | FALSE | yes | |
| rainy | cool | normal | FALSE | yes | |
| rainy | cool | normal | TRUE | no | |
| overcast | cool | normal | TRUE | yes | |
| sunny | mild | high | FALSE | no | |
| sunny | cool | normal | FALSE | yes | |
| rainy | mild | normal | FALSE | yes | |
| sunny | mild | normal | TRUE | yes | no |
| overcast | mild | high | TRUE | yes | yes |
| overcast | hot | normal | FALSE | yes | yes |
| rainy | mild | high | TRUE | no | yes |

## What is a good Classifier? III

4 test instances; 2 correct predictions, 2 incorrect predictions

$$\begin{aligned}\text{Accuracy} &= \frac{\text{Number of correctly labelled test instances}}{\text{Total number of test instances}} \\ &= \frac{2}{4} = 50\%\end{aligned}$$

## Evaluation Metrics

For a two class problem, assume an **Interesting** class (`I`) and an **Uninteresting** class (`U`).

A classifier may classify

- an Interesting instance as `I` (True Positive, TP)
- an Interesting instance as `U` (False Negative, FN)
- an Uninteresting instance as `I` (False Positive, FP)
- a Uninteresting instance as `U` (True Negative, TN)

| | | ***Predicted*** | |
|---|---|---|---|
| | | ***I*** | ***U*** |
| ***Actual*** | ***I*** | true positive (TP) | false negative (FN) |
| | ***U*** | false positive (FP) | true negative (TN) |

## Accuracy

- **Classification accuracy** is the proportion of instances for which we have correctly predicted the label, which corresponds to:

$$\mathrm{ACC} = \frac{TP + TN}{TP + FP + FN + TN}$$

n.b. Independent of `I` and `U`

- Alternatively, we sometimes talk about the **error rate**:

$$\mathrm{ER} = \frac{\mathit{FP} + \mathit{FN}}{\mathit{TP} + \mathit{FP} + \mathit{FN} + \mathit{TN}}$$

  N.B. $\mathrm{ER} = 1 - \mathrm{ACC}$

- We also sometimes refer to the **error rate reduction**, comparing the error rate $\mathrm{ER}$ for a given method with that for an alternative method $\mathrm{ER}_0$:

$$\mathrm{ERR} = \frac{\mathrm{ER}_0 - \mathrm{ER}}{\mathrm{ER}_0}$$

## Precision and Recall I

- With respect to **just the interesting class**:
- **Precision**: How often are we correct, when we predict that an instance is interesting?
- **Recall**: What proportion of the truly interesting instances have we correctly identified as interesting?

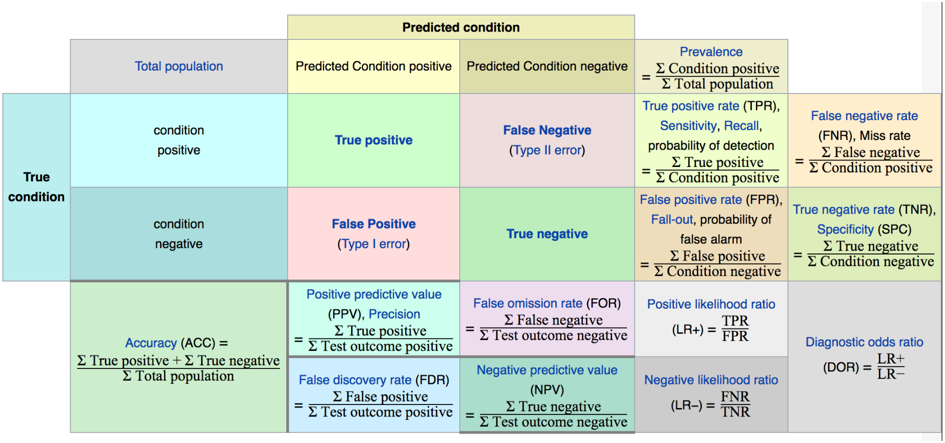
$$\begin{aligned}\text{Precision} &= \frac{TP}{TP + FP} \\ \text{Recall} &= \frac{TP}{TP + FN}\end{aligned}$$

## Precision and Recall II

- Precision/Recall are typically in an inverse relationship. We can generally set up our classifier, so that:
  - The classifier has high Precision, but low Recall
  - The classifier has high Recall, but low Precision
- But, we want **both** Precision and Recall to be high. A popular metric that evaluates this is **F-score**:

$$\begin{aligned} F_\beta &= \frac{(1+\beta^2)PR}{\beta^2 P + R} \\ F_1 &= \frac{2PR}{P+R} \end{aligned}$$

## Many other Metrics!



| | | Predicted condition | | | |
|---|---|---|---|---|---|
| | Total population | Predicted Condition positive | Predicted Condition negative | Prevalence $= \frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$ | |
| True condition | condition positive | **True positive** | **False Negative** (Type II error) | True positive rate (TPR), Sensitivity, Recall, probability of detection $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$ | False negative rate (FNR), Miss rate $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$ |
| | condition negative | **False Positive** (Type I error) | **True negative** | False positive rate (FPR), Fall-out, probability of false alarm $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$ | True negative rate (TNR), Specificity (SPC) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$ |
| | Accuracy (ACC) = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$ | Positive predictive value (PPV), Precision $= \frac{\Sigma \text{ True positive}}{\Sigma \text{ Test outcome positive}}$ | False omission rate (FOR) $= \frac{\Sigma \text{ False negative}}{\Sigma \text{ Test outcome negative}}$ | Positive likelihood ratio (LR+) $= \frac{\text{TPR}}{\text{FPR}}$ | Diagnostic odds ratio (DOR) $= \frac{\text{LR+}}{\text{LR−}}$ |
| | | False discovery rate (FDR) $= \frac{\Sigma \text{ False positive}}{\Sigma \text{ Test outcome positive}}$ | Negative predictive value (NPV) $= \frac{\Sigma \text{ True negative}}{\Sigma \text{ Test outcome negative}}$ | Negative likelihood ratio (LR–) $= \frac{\text{FNR}}{\text{TNR}}$ | |

From Wikipedia:

`https://en.wikipedia.org/wiki/Sensitivity_and_specificity`

## Multi-class Evaluation I

For a multi-class problem, assume an **Interesting** class (`I`) and several **Uninteresting** classes (`U1`, `U2`, ...).

| | | *Predicted* | | | |
|---|---|---|---|---|---|
| | | ***I*** | ***U1*** | ***U2*** | ... |
| | ***I*** | TP | FN | FN | ... |
| ***Actual*** | ***U1*** | FP | TN | TN | ... |
| | ***U2*** | FP | TN | TN | ... |
| | ... | ... | ... | ... | ... |

This is a **Confusion Matrix**.
Typically, all classes are “interesting” in a multi-class context.

## Multi-class Evaluation II

- Note that the natural definition of Accuracy still makes sense in a multi-class context, but the technical definition behaves strangely (re-interprets the multi-class problem as a special case of a two-class problem, aka One-vs-Rest)
- Precision/Recall/F-Score are all calculated **per-class**, and must be averaged across $c$ classes:
    - **macro-averaging**:
      calculate P, R per class and then take mean

$$\begin{aligned} \text{Precision}_M &= \frac{\sum_{i=1}^{c} \text{Precision}(i)}{c} \\ \text{Recall}_M &= \frac{\sum_{i=1}^{c} \text{Recall}(i)}{c} \end{aligned}$$

## Multi-class Evaluation III

- Precision/Recall/F-Score are all calculated **per-class**, and must be averaged:
  - **micro-averaging**:
    combine all test instances into a single pool

$$\begin{aligned}\text{Precision}_\mu &= \frac{\sum_{i=1}^{c} TP_i}{\sum_{i=1}^{c} TP_i + FP_i}\\ \text{Recall}_\mu &= \frac{\sum_{i=1}^{c} TP_i}{\sum_{i=1}^{c} TP_i + FN_i}\end{aligned}$$

## Multi-class Evaluation IV

- Precision/Recall/F-Score are all calculated **per-class**, and must be averaged:
  - **weighted averaging**:
    calculate P, R per class and then take weighted mean, based on the proportion of instances in that class

$$\begin{aligned}\text{Precision}_W &= \sum_{i=1}^{c}(\frac{n_i}{N})\text{Precision}(i)\\ \text{Recall}_W &= \sum_{i=1}^{c}(\frac{n_i}{N})\text{Recall}(i)\end{aligned}$$

## Multi-class Evaluation V

- Most striking differences occur when “don’t know” is permitted as a prediction
- Small differences can occur, depending on when averaging takes place:
    - For example, is “macro-averaged F-score” the F-score of macro-averaged P (over classes) and macro-averaged R (over classes)? or the macro-average (over classes) of the F-score for each class?
    - If we are doing Repeated Random Subsampling, and want “weighted-averaged Precision”, do we average the weighted Precision (over classes) for each iteration of Random Subsampling? or do we take the weighted average (over classes) of the Precision averaged over the iterations of Subsampling? or the weighted average over the instances aggregated over the iterations?

## Lecture Outline

## Baselines vs. Benchmarks

- **Baseline** = naive method which we would expect any reasonably well-developed method to better

  *e.g. for a novice marathon runner, the time to **walk** 42km*

- **Benchmark** = established rival technique which we are pitching our method against

  *e.g. for a marathon runner, the time of our last marathon run/the world record time/3 hours/...*

- "Baseline" often used as umbrella term for both meanings

## The Importance of Baselines

- Baselines are important in establishing whether any proposed method is doing better than "dumb and simple"

  *"dumb" methods often work surprisingly well*

- Baselines are valuable in getting a sense for the intrinsic difficulty of a given task (cf. accuracy = 5% vs. 99%)
- In formulating a baseline, we need to be sensitive to the importance of positives and negatives in the classification task

  *limited utility of a baseline of `unsuitable` for a classification task aimed at detecting potential sites for new diamond mines (as nearly all sites are unsuitable)*

## Random Baseline

**Method 1:** randomly assign a class to each test instance

- Often the only option in unsupervised/semi-supervised contexts

**Method 2:** randomly assign a class $c_k$ to each test instance, weighting the class assignment according to $P(c_k)$

- Assumes we know the class prior probabilities
- Alleviate effects of variance by:
  - running method $N$ times and calculating the mean accuracy

    *OR*
  - arriving at a deterministic estimate of the accuracy of random assignment $= \sum_i P(c_i)^2$

## Zero-R

- **Method:** classify all instances according to the most common class in the training data
- The most commonly used baseline in machine learning
- Also known as **majority class** baseline
- Inappropriate if the majority class is `FALSE` and the learning task is to identify needles in the haystack
- For `weather.nominal`, zero-R class = `yes`

## One-R (One Rule)

Creates one rule for each attribute in the training data, then selects the rule with the smallest error rate as its "one rule"

- **Method:** create a "decision stump" for each attribute, with branches for each value, and populate the leaf with the majority class at that leaf; select the decision stump which leads to the lowest error rate over the training data

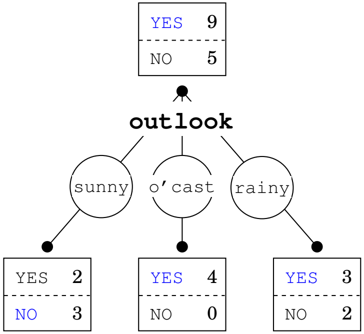
## One-R pseudo-code

```
 For each attribute,
  For each value of the attribute, make a rule:
   (i) count how often each class appears
  (ii) find the most frequent class
 (iii) make the rule assign that class to this value
  Calculate the error rate of the rules
Choose the attribute whose rules produce the smallest
error rate
```

## Weather dataset

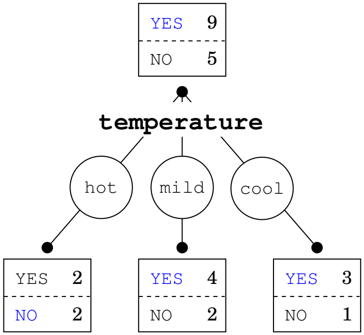
| Outlook | Temperature | Humidity | Windy | Play |
|---|---|---|---|---|
| sunny | hot | high | FALSE | no |
| sunny | hot | high | TRUE | no |
| overcast | hot | high | FALSE | yes |
| rainy | mild | high | FALSE | yes |
| rainy | cool | normal | FALSE | yes |
| rainy | cool | normal | TRUE | no |
| overcast | cool | normal | TRUE | yes |
| sunny | mild | high | FALSE | no |
| sunny | cool | normal | FALSE | yes |
| rainy | mild | normal | FALSE | yes |
| sunny | mild | normal | TRUE | yes |
| overcast | mild | high | TRUE | yes |
| overcast | hot | normal | FALSE | yes |
| rainy | mild | high | TRUE | no |

## Decision Stump (outlook)



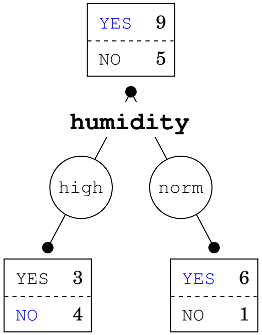
Total errors = $\frac{4}{14}$

## Decision Stump (temperature)



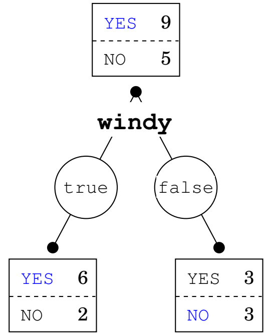
Total errors = $\frac{5}{14}$

## Decision Stump (humidity)



Total errors = $\frac{4}{14}$

# Decision Stump (windy)



Total errors = $\frac{5}{14}$

## One-R: Reflections

- Advantages:
  - simple to understand and implement
  - simple to comprehend the results
  - surprisingly good results
- Disadvantages:
  - unable to capture attribute interactions
  - bias towards high-arity attributes (attributes with many possible values)

## Summary

- How do we set up an evaluation of a classification system?
- What are the measures we use to assess the performance of the classification system?
- What is a baseline? What are some examples of reasonable baselines to compare with?

## References I

Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining*. Addison Wesley, 2006.

David Wolpert and William Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1:67–82, 1997.