

COMP30120

Evaluation in Machine Learning Part 2

Derek Greene

**School of Computer Science and Informatics
Autumn 2015**



Overview

- Part 1
 - Objectives of Evaluation
 - A/B Testing
 - Statistical Significance
 - Student's t-test
 - Tests for proportions
- Part 2
 - Evaluation Measures
 - Overfitting
 - Experimental Setup

Evaluation

When making predictions, we need some kind of measure to capture how often the model makes correct or incorrect predictions, and how severe the mistakes are.

Misclassification Rate:











Fraction of incorrect predictions made by the classifier.

$$MR = \frac{\# \text{ incorrect predictions}}{\text{total predictions}}$$

Accuracy:

Fraction of correct predictions made by the classifier.

$$ACC = \frac{\# \text{ correct predictions}}{\text{total predictions}}$$

| Email | Label | Prediction | Correct? |
|-------|----------|------------|---|
| 1 | spam | non-spam |  |
| 2 | spam | spam |  |
| 3 | non-spam | non-spam |  |
| 4 | spam | spam |  |
| 5 | non-spam | spam |  |
| 6 | non-spam | non-spam |  |
| 7 | spam | spam |  |
| 8 | non-spam | spam |  |
| 9 | non-spam | non-spam |  |
| 10 | spam | spam |  |

$$MR = \frac{3}{10} = 0.3 \quad ACC = \frac{7}{10} = 0.7$$

Example: Hotel Reviews

Q. Can we predict the “helpfulness” of TripAdvisor hotel reviews?

The screenshot displays the TripAdvisor interface for 'Golf Hotel Bled'. It features a summary section with an 86% recommendation rate and 52 reviews, broken down by rating (Excellent: 21, Very good: 21, Average: 5, Poor: 3, Terrible: 2). Below this, three individual reviews are shown, each with a circled helpfulness count:

- Review 1:** User 'ki_holland23' (London) from August 21, 2008, rated 5 stars. The review text describes a honeymoon package. The helpfulness count is '2/2 found this review helpful'.
- Review 2:** User 'singold' (NJ) from May 22, 2007, rated 5 stars. The review text describes a 'fairyland location'. The helpfulness count is '1/3 found this review helpful'.
- Review 3:** User 'A TripAdvisor Member' (england) from July 24, 2005, rated 4 stars. The review text describes the hotel's location and amenities. The helpfulness count is '9/11 found this review helpful'.

Each review includes a 'Was this review helpful?' prompt with 'Yes' and 'No' buttons, and links for 'View profile', 'Send message', 'Compliment review', and 'Report inappropriate content'. The bottom of the page shows the user's ratings for the hotel across various categories like Value, Rooms, Location, Cleanliness, Check in / front desk, and Service.

“Learning to Recommend Helpful Hotel Reviews”

O’Mahony & Smyth, 3rd ACM RecSys Conference, 2009

<http://dl.acm.org/citation.cfm?id=1639774>

Example: Hotel Reviews

- Compare performance of Naïve Bayes and Support Vector Machine (SVM) classifiers on review data using Weka.
- Test set: 105 “Helpful”, 60 “Unhelpful” reviews
- Testing option: Hold-out validation with 66/33% hold-out split.

```
Correctly Classified Instances      117          70.9091 %
Incorrectly Classified Instances    48          29.0909 %
Kappa statistic                    0.3071
Mean absolute error                0.2909
Root mean squared error            0.5394
Relative absolute error            62.6804 %
Root relative squared error        112.1168 %
Total Number of Instances         165
```

=== Detailed Accuracy By Class ===

| | TP Rate | FP Rate | Precision | Recall | F-Measure | ROC Area | Class |
|---------------|---------|---------|-----------|--------|-----------|----------|-------|
| | 0.895 | 0.617 | 0.718 | 0.895 | 0.797 | 0.639 | good |
| | 0.383 | 0.105 | 0.676 | 0.383 | 0.489 | 0.639 | bad |
| Weighted Avg. | 0.709 | 0.431 | 0.703 | 0.709 | 0.685 | 0.639 | |

=== Confusion Matrix ===

| a | b | <-- classified as |
|----|----|-------------------|
| 94 | 11 | a = good |
| 37 | 23 | b = bad |

SVM

```
Correctly Classified Instances      103          62.4242 %
Incorrectly Classified Instances    62          37.5758 %
Kappa statistic                    0.1995
Mean absolute error                0.3793
Root mean squared error            0.5316
Relative absolute error            81.7353 %
Root relative squared error        110.5048 %
Total Number of Instances         165
```

=== Detailed Accuracy By Class ===

| | TP Rate | FP Rate | Precision | Recall | F-Measure | ROC Area | Class |
|---------------|---------|---------|-----------|--------|-----------|----------|-------|
| | 0.686 | 0.483 | 0.713 | 0.686 | 0.699 | 0.674 | good |
| | 0.517 | 0.314 | 0.484 | 0.517 | 0.5 | 0.674 | bad |
| Weighted Avg. | 0.624 | 0.422 | 0.63 | 0.624 | 0.627 | 0.674 | |

=== Confusion Matrix ===

| a | b | <-- classified as |
|----|----|-------------------|
| 72 | 33 | a = good |
| 29 | 31 | b = bad |

Naïve
Bayes

Example: Hotel Reviews

- Compare performance of Naïve Bayes and Support Vector Machine (SVM) classifiers on review data using Weka.
- Test set: 105 “Helpful”, 60 “Unhelpful” reviews
- Testing option: Hold-out validation with 66/33% hold-out split.

SVM

Accuracy = 70.9%

| Prediction | | |
|------------|----|---|
| H | U | |
| 94 | 11 | H |
| 37 | 23 | U |

Real World

SVM biased toward majority class (H)

Naïve Bayes

Accuracy = 62.4%

| Prediction | | |
|------------|----|---|
| H | U | |
| 72 | 33 | H |
| 29 | 31 | U |

Real World

What if this is important?

Evaluation Measures

Confusion matrix summarises the performance of an algorithm, when compared with the real classes (“ground truth”).

| | | Predicted Class | |
|------------|----------|--|---|
| | | Positive | Negative |
| Real Class | Positive | TP True positive Correct! | FN False Negative (Type II error) |
| | Negative | FP False Positive (Type I error) | TN True Negative Correct! |











Clinical Example: Predict a case as *positive* (person has the disease) or *negative* (person does not have the disease)

- **TP** = Sick people correctly predicted as sick
- **FP** = Healthy people incorrectly predicted as sick
- **TN** = Healthy people correctly predicted as healthy
- **FN** = Sick people incorrectly predicted as healthy

Evaluation Measures

Spam Filtering Example: Predict emails as *spam* or *non-spam*...

- **TP** = Spam emails correctly predicted as spam
- **FP** = Non-spam emails incorrectly predicted as spam
- **TN** = Non-spam emails correctly predicted as non-spam
- **FN** = Spam emails incorrectly predicted as non-spam

| Email | Label | Prediction | Correct? | Outcome |
|-------|----------|------------|---|---------|
| 1 | spam | non-spam |  | FN |
| 2 | spam | spam |  | TP |
| 3 | non-spam | non-spam |  | TN |
| 4 | spam | spam |  | TP |
| 5 | non-spam | spam |  | FP |
| 6 | non-spam | non-spam |  | TN |
| 7 | spam | spam |  | TP |
| 8 | non-spam | spam |  | FP |
| 9 | non-spam | non-spam |  | TN |
| 10 | spam | spam |  | TP |

| Predicted Class | | |
|-----------------|------|------|
| Spam | Non | |
| TP=4 | FN=1 | Spam |
| FP=2 | TN=3 | Non |

Real Class

Evaluation Measures

- **Accuracy:** Fraction of predictions correct

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

- **True Positive Rate:**
Focus on TPs
Also called *Sensitivity*

$$\text{TPRate} = \frac{TP}{TP + FN}$$

| | | Predicted | | |
|--------|-----|-----------|-----|------|
| | | Pos | Neg | |
| P N | Pos | TP | FN | Pos |
| | Neg | FP | TN | Neg |
| | | | | Real |

- **False Positive Rate:**
Focus on FPs

$$\text{FPRate} = \frac{FP}{FP + TN}$$

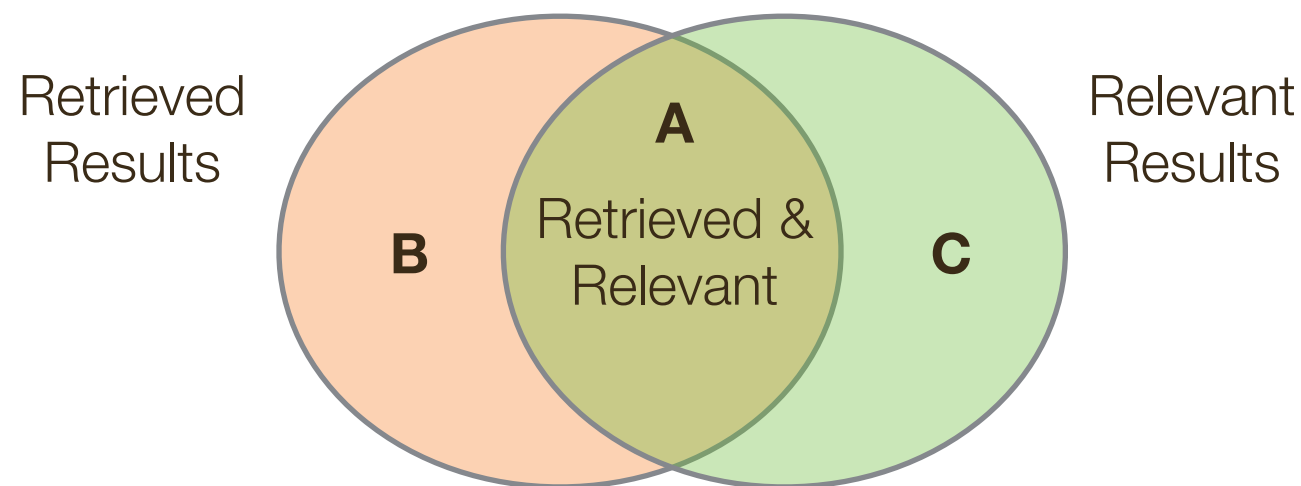
| | | Predicted | | |
|--------|-----|-----------|-----|------|
| | | Pos | Neg | |
| P N | Pos | TP | FN | Pos |
| | Neg | FP | TN | Neg |
| | | | | Real |

- **True Negative Rate:**
Focus on TNs
Also called *Specificity*

$$\text{TNRate} = \frac{TN}{FP + TN}$$

Precision & Recall

- Measures from information retrieval, but used in ML evaluation.
- **Precision**: proportion of retrieved results that are relevant.
- **Recall**: proportion of relevant results that are retrieved.



$$\text{Precision} = \frac{A}{A + B}$$

$$\text{Recall} = \frac{A}{A + C}$$

Search Example: Given a collection of 100k documents, we want to find all documents on “water charges”. In fact, 45 relevant documents actually exist.

Perform a search, 9 out of 10 results on first page are relevant documents.

➡ Precision = $9/10 = 90\%$ of retrieved results were relevant.

➡ Recall = $9/45 = 20\%$ of all possible relevant results were retrieved.

Precision & Recall

- Precision & Recall also used as general measures to evaluate machine learning algorithms.

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN} = \text{Sensitivity}$$

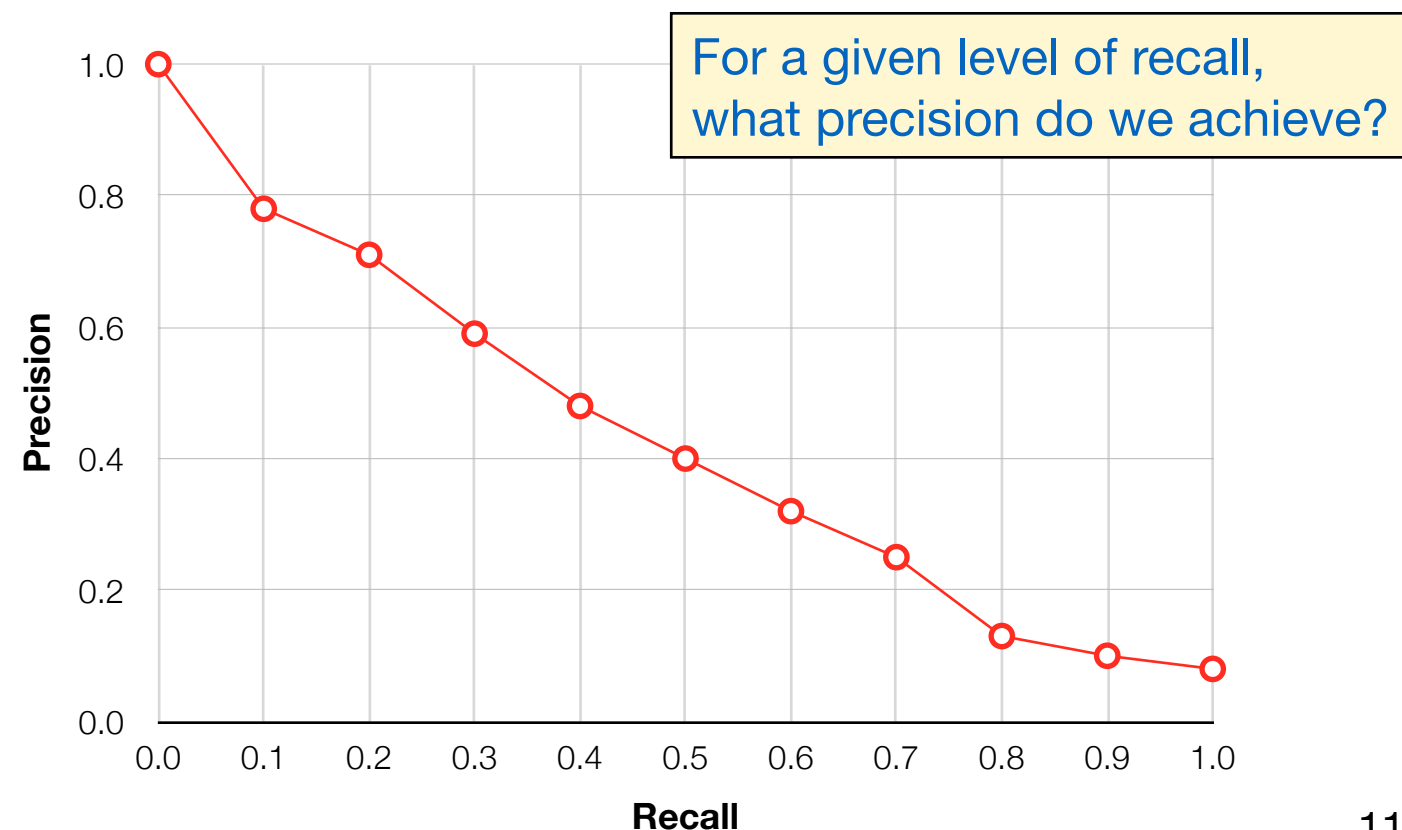
| | | Predicted | | |
|---|--|-----------|-----|-----|
| | | Pos | Neg | |
| P | | TP | FN | Pos |
| N | | FP | TN | Neg |

Real

| | | Predicted | | |
|---|--|-----------|-----|-----|
| | | Pos | Neg | |
| P | | TP | FN | Pos |
| N | | FP | TN | Neg |

Real

- Plot the trade-off between the two measures using a **Precision-Recall (PR) curve**.
- Used to study the output of a binary classifier.
- Measure precision at fixed recall intervals.



Example Calculations

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

$$= \frac{4 + 3}{10} = 0.7$$











$$\text{TPRate} = \frac{TP}{TP + FN} = \frac{4}{4 + 1} = 0.8$$

$$\text{FPRate} = \frac{FP}{FP + TN} = \frac{2}{2 + 3} = 0.4$$

$$\text{TNRate} = \frac{TN}{FP + TN} = \frac{3}{2 + 3} = 0.6$$

$$\text{Precision} = \frac{TP}{TP + FP} = \frac{4}{4 + 2} = 0.667$$

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{4}{4 + 1} = 0.8$$

| | Label | Prediction | Correct? | Outcome |
|----|----------|------------|---|---------|
| 1 | spam | non-spam |  | FN |
| 2 | spam | spam |  | TP |
| 3 | non-spam | non-spam |  | TN |
| 4 | spam | spam |  | TP |
| 5 | non-spam | spam |  | FP |
| 6 | non-spam | non-spam |  | TN |
| 7 | spam | spam |  | TP |
| 8 | non-spam | spam |  | FP |
| 9 | non-spam | non-spam |  | TN |
| 10 | spam | spam |  | TP |

Predicted Class

| Spam | Non | |
|-------------|-------------|------|
| TP=4 | FN=1 | Spam |
| FP=2 | TN=3 | Non |

Real Class

Balanced Accuracy

- Skewed class distributions occur when one class is over-represented in the data.
 - e.g. Fraud detection: Vast majority of financial transactions are legitimate, a small fraction are fraudulent.
 - Other examples: medical diagnosis, e-commerce, security.

- High accuracy can be achieved by biased (or trivial) classifiers.

⇒ Accuracy = 90%

| Classified as | | | |
|---------------|-----|-----|------------|
| Pos | Neg | | |
| 0 | 10 | Pos | Real World |
| 0 | 90 | Neg | |

- To deal with skewed classes, use a balanced evaluation measure. Measures include:
 - **Balance Accuracy Rate (BAR)**: Mean of TP Rate and TN Rate
 - **Balance Error Rate (BER)**: Mean of FP Rate and FN Rate

Balanced Accuracy

- **F-Measure**: A single measure that trades off precision against recall, for a given level of balance.

$$F = \frac{(1 + \beta^2) \times \text{Precision} \times \text{Recall}}{\beta^2 \times \text{Precision} + \text{Recall}}$$

- The Beta parameter controls the trade-off:
 - $\beta < 1$ Focus more on Precision
 - $\beta = 1$ Harmonic mean of Precision and Recall.
 - $\beta > 1$ Focus more on Recall
- **F1-Measure**: Most widely-used variant, sets $\beta = 1$

$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad \text{harmonic mean of precision and recall}$$

ROC Analysis

- Many classifiers produce a prediction score, which is then converted to a binary decision based on some **decision threshold** θ
- Changing this threshold value can lead to different predictions, and so a different confusion matrix.

| | Label | Score | > 0.5? | Prediction | Outcome |
|----|----------|-------|--------|------------|---------|
| 1 | spam | 0.1 | N | non-spam | FN |
| 2 | spam | 0.8 | Y | spam | TP |
| 3 | non-spam | 0.6 | Y | spam | FP |
| 4 | spam | 0.9 | Y | spam | TP |
| 5 | non-spam | 0.8 | Y | spam | FP |
| 6 | non-spam | 0.2 | N | non-spam | TN |
| 7 | spam | 0.8 | Y | spam | TP |
| 8 | non-spam | 0.6 | Y | spam | FP |
| 9 | non-spam | 0.1 | N | non-spam | TN |
| 10 | spam | 0.9 | Y | spam | TP |

Decision
Threshold

$$\theta = 0.5$$

| Predicted Class | | |
|-----------------|------|------|
| Spam | Non | |
| TP=4 | FN=1 | Spam |
| FP=3 | TN=2 | Non |

Real Class

| | Label | Score | > 0.7? | Prediction | Outcome |
|----|----------|-------|--------|------------|---------|
| 1 | spam | 0.1 | N | non-spam | FN |
| 2 | spam | 0.8 | Y | spam | TP |
| 3 | non-spam | 0.6 | N | non-spam | TN |
| 4 | spam | 0.9 | Y | spam | TP |
| 5 | non-spam | 0.8 | Y | spam | FP |
| 6 | non-spam | 0.2 | N | non-spam | TN |
| 7 | spam | 0.8 | Y | spam | TP |
| 8 | non-spam | 0.6 | N | non-spam | TN |
| 9 | non-spam | 0.1 | N | non-spam | TN |
| 10 | spam | 0.9 | Y | spam | TP |

Decision
Threshold

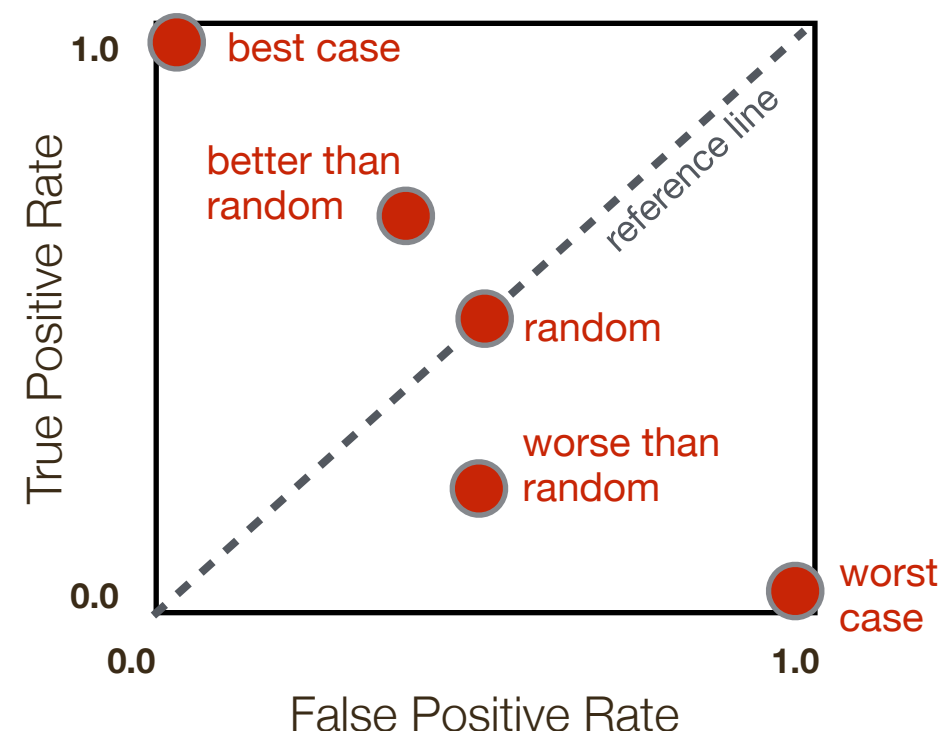
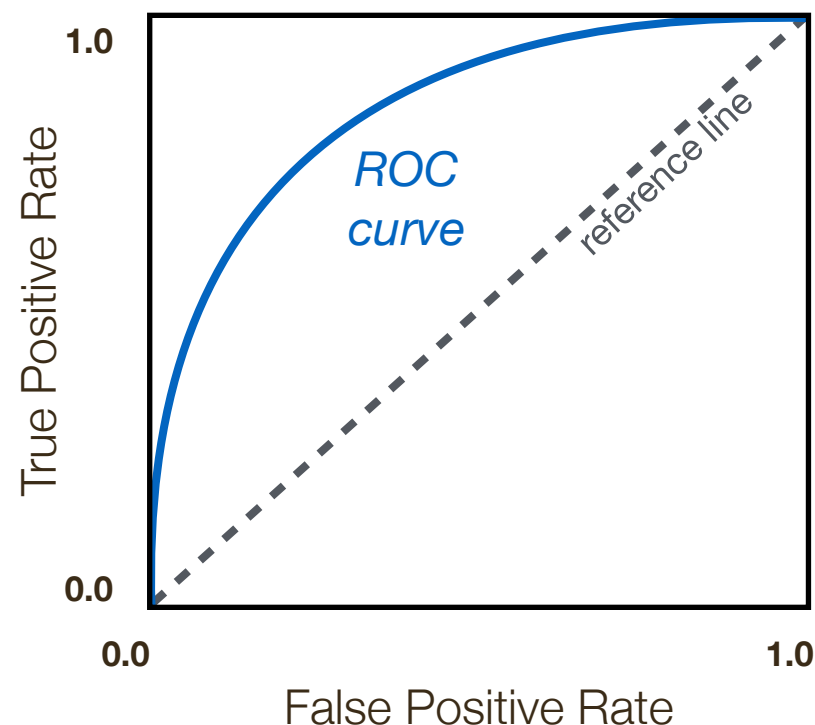
$$\theta = 0.7$$

| Predicted Class | | |
|-----------------|------|------|
| Spam | Non | |
| TP=4 | FN=1 | Spam |
| FP=1 | TN=4 | Non |

Real Class

ROC Analysis

- Often want to compare the performance of classifiers at many different decision thresholds (i.e. summarise many confusion matrices).
- A **Receiver Operating Characteristic (ROC Curve)** is a graphical plot of how the true positive rate and false positive rate change over many different thresholds. The curve is drawn by plotting a point for each feasible threshold and joining them.
- A trained classifier should always be above the “random” reference line. The strength of the classifier increases as the ROC curve moves further from the line (i.e. closer to top left corner).

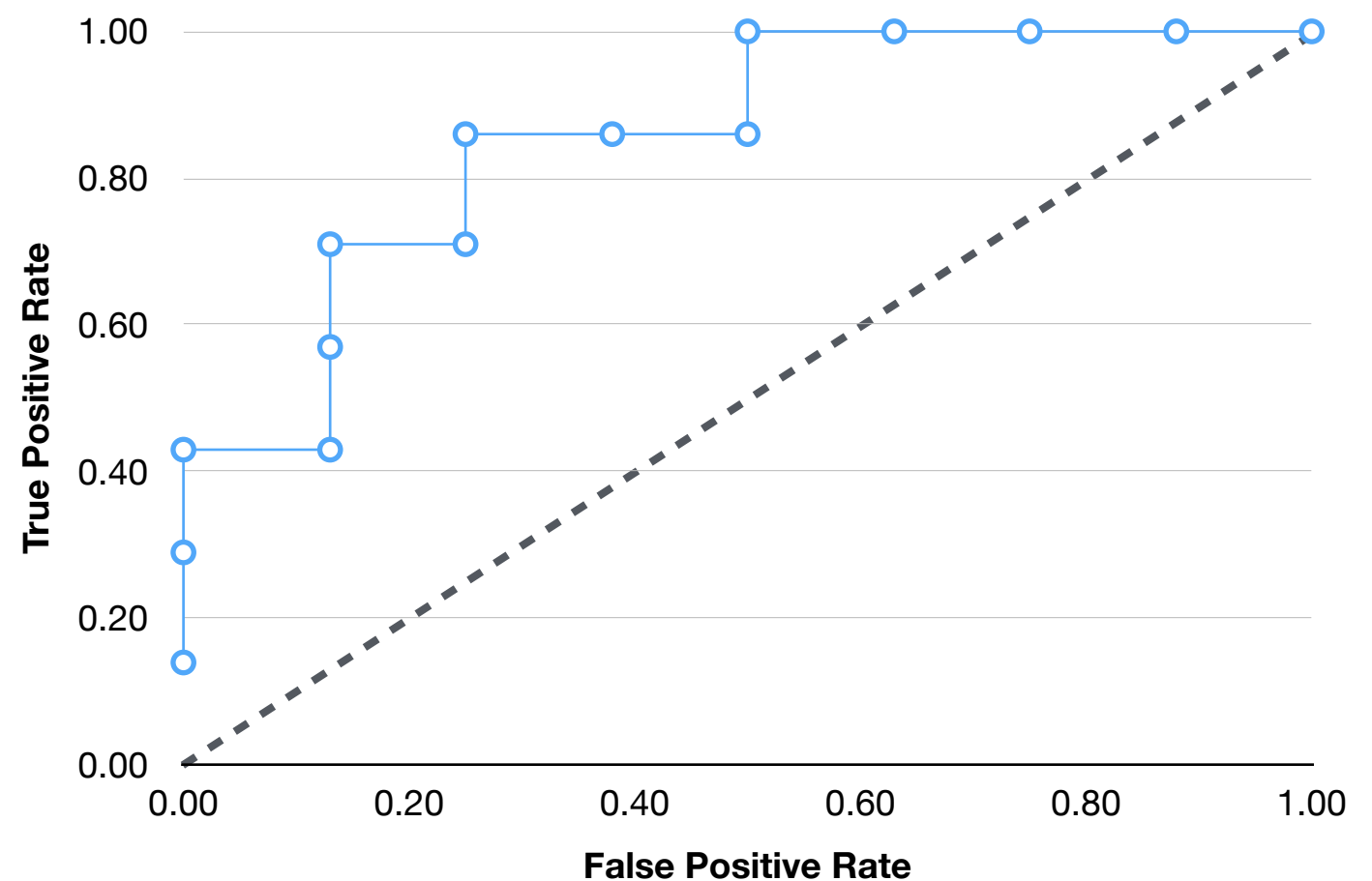


Example: ROC Analysis

- Given a ranking classifier which will score test samples with a probability P of belonging to the positive class.
- Decision threshold θ controls whether a sample will be classified as positive or negative - i.e. $P > \theta$

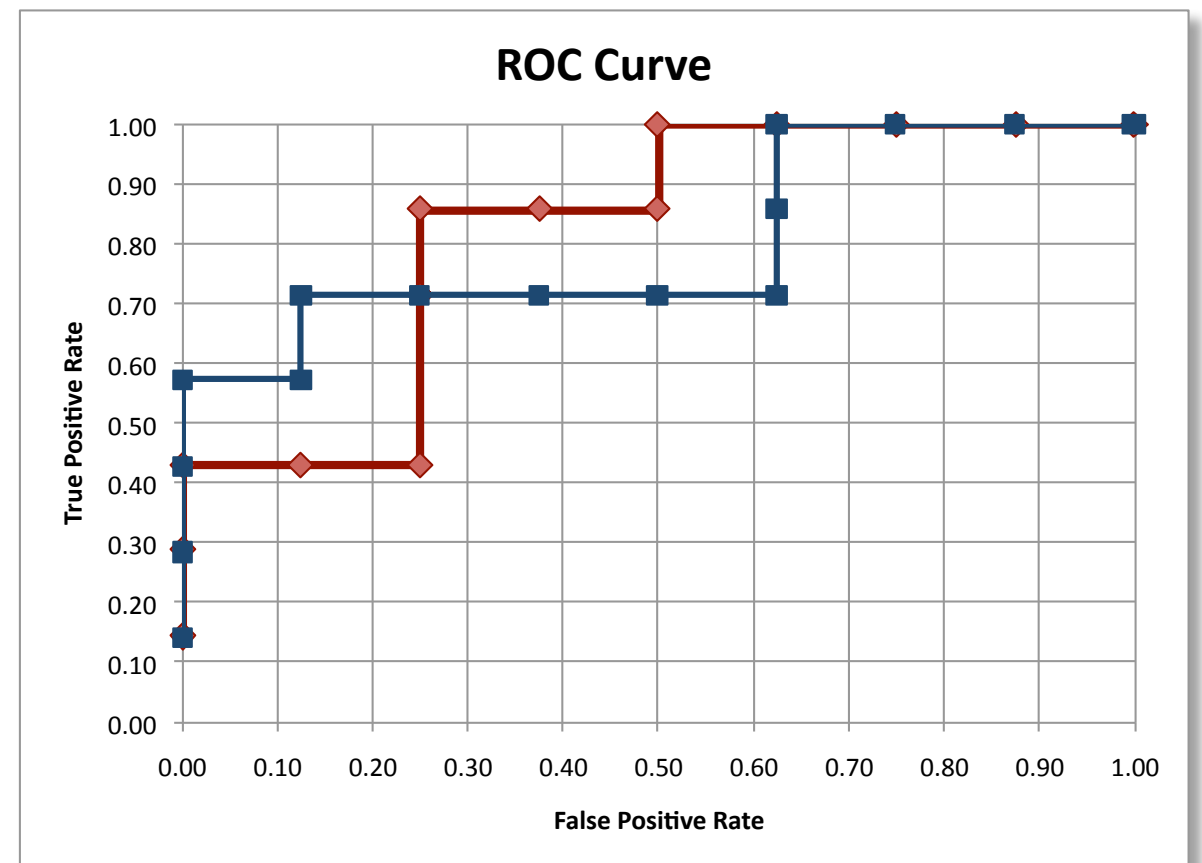
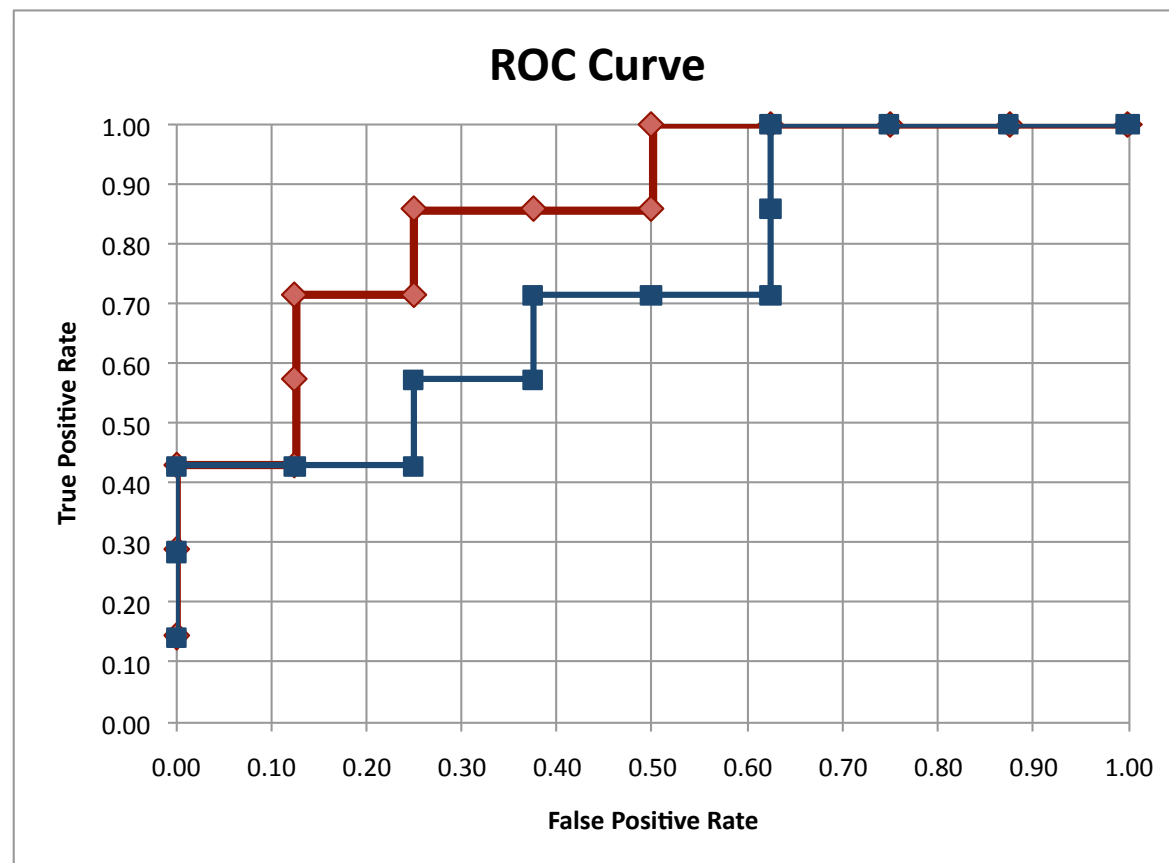
Single example $\theta = 0.5$

| P | > 0.5 | Real |
|------|-------|------|
| 0.99 | 1 | 1 |
| 0.90 | 1 | 1 |
| 0.80 | 1 | 1 |
| 0.85 | 1 | 0 |
| 0.70 | 1 | 1 |
| 0.70 | 1 | 1 |
| 0.65 | 1 | 0 |
| 0.60 | 1 | 1 |
| 0.45 | 0 | 0 |
| 0.45 | 0 | 0 |
| 0.40 | 0 | 1 |
| 0.30 | 0 | 0 |
| 0.20 | 0 | 0 |
| 0.20 | 0 | 0 |
| 0.20 | 0 | 0 |



Comparing ROC Curves

- Often want to compare the performance of two classifiers at different thresholds → we can look at their ROC curves.
- In some cases, one classifier will always be better than another across all values of θ . In other cases, it will be more complicated...



- ➡ Make comparisons based on **Area Under the Curve (AUC)**. A better classifier will have a ROC curve closer to top-left corner, giving a larger area under the curve.

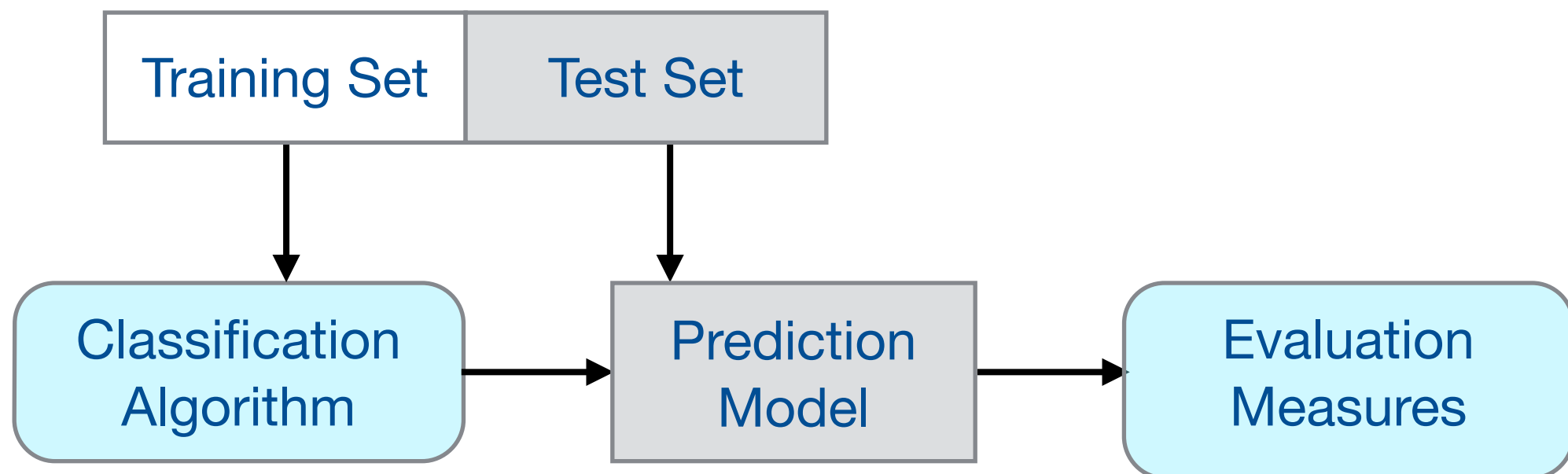
Overfitting

- For real-world tasks, we are interested in **generalisation accuracy**.
- **Overfitting**: Model is fitted too closely to the training data (including its noise). The model cannot generalise to situations not presented during training, so it is not useful when applied to unseen data.
- **Possible Causes**
 - *Small training set*: Classifier only given a few examples, may not be representative of the underlying concepts.
 - *Complex model*: Model has too many parameters relative to the number of training examples.
 - *Noise*: Spurious or contradictory patterns in the training data.
 - *High-dimensionality*: Data has many irrelevant features (dimensions) containing noise which leads to a poor model.
- ➔ A good model must not only fit the training data well, but also accurately classify examples that it has never seen before.

Experimental Setup

- **Simple Hold-Out Strategy:**

- Keep some training data back (the **hold-out set**) to use for evaluating the model produced by the classifier.
- Use performance on the hold-out set as a proxy for performance on unseen data (i.e. generalisation accuracy).



- Using a hold-out set avoids **peeking** - when the performance of a model is evaluated using the same data used to train it.
e.g. Use of training data for testing in Weka can produce unrealistic accuracy results that are “too good to be true”.

Experimental Setup

- **Random Split:**

- Randomly assign all examples to training/test subsets.

| | |
|-----------------------|-------------------|
| Training Set (50%) | Test Set (50%) |
| Training Set (66%) | Test Set (33%) |
| Training Set (80%) | Test (20%) |

- Problem: Each random split might give different results.

- **Multiple Test Splits:**

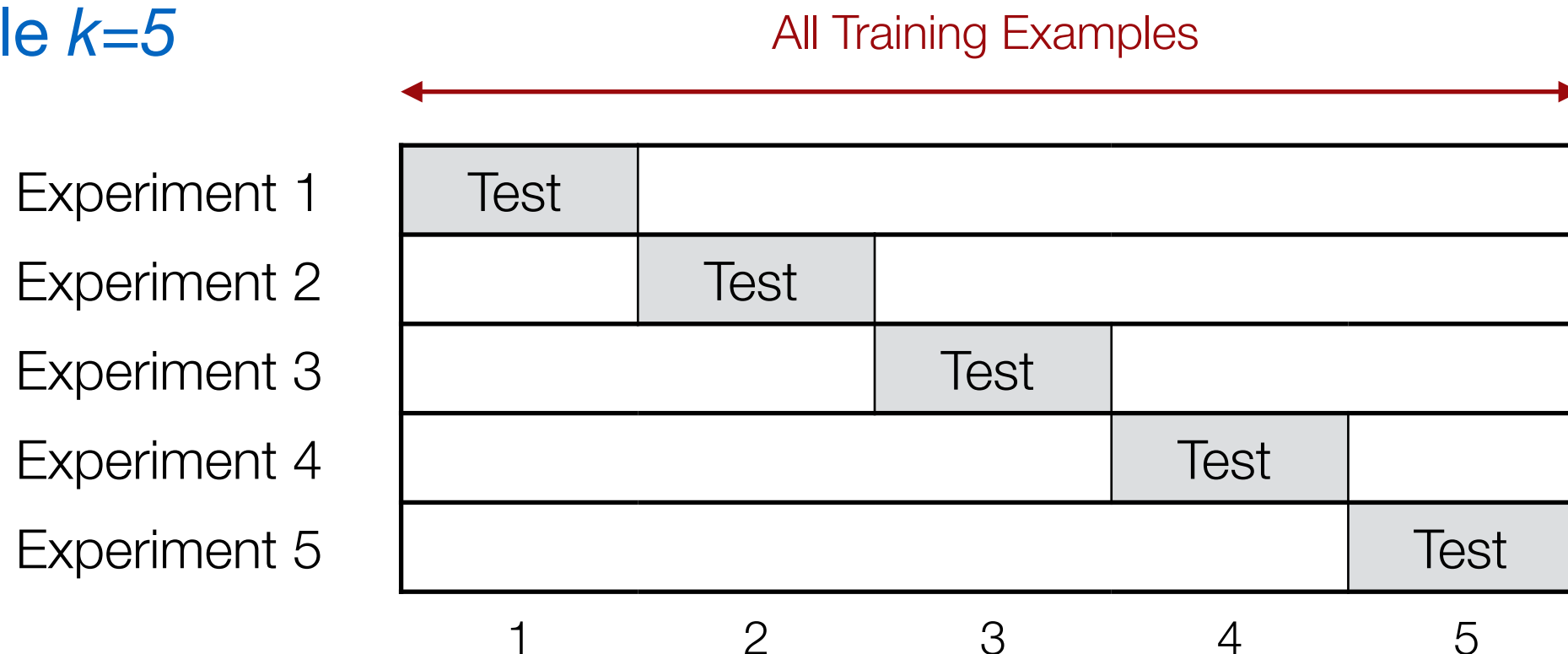
- Reduce the variance of the random process by repeating the random split many times and calculate average accuracy/error.
- Problem: Some examples will never included for training or testing, while others may be selected multiple times.

Cross Validation

- ***k*-Fold Cross Validation:**

- Divide the data into k disjoint subsets - “folds” (e.g. $k=10$).
- For each of k experiments, use $k-1$ folds for training and the selected one fold for testing .
- Repeat for all k folds, average the accuracy/error rates.

Example $k=5$



Experimental Setup

- **Three-Way Hold-Out Strategy:** Divide the full dataset into three different subsets.
 1. **Training set:** The subset of examples used for learning.
 2. **Validation set:** The subset of examples used to tune the classifier (e.g. select parameter values).
 3. **Test set:** The subset of examples used only to assess the performance of a fully-trained classifier.

| | | |
|-----------------------|-------------------------|-------------------|
| Training Set (50%) | Validation Set (20%) | Test Set (30%) |
| Training Set (40%) | Validation Set (20%) | Test Set (40%) |

- ➔ This avoids a bias in evaluation of the model, where reusing examples from the validation set could lead to underestimates of the real error rate.

Comparing Classifiers

- Robust evaluation process: Apply k -fold cross validation with a three-way hold-out strategy.

Overall Process

- Divide data set into k folds.
 - FOR EACH of the k folds:
 - Create test set T from the k -th fold.
 - Create training set R from the remaining examples.
 - Divide R into R_1 and validation set V .
 - FOR EACH classifier
 - * Use V to tune parameters on a model trained with R_1 .
 - * Use selected parameters to train a model with R .
 - * Measure Accuracy on T .
 - Collate results, assess significance of differences.
-
- Significance of proportions of wins and losses across all experiments can be measured statistically (e.g. McNemar's test).
 - We can repeat entire process multiple times to further reduce random variance - e.g. 10 x 10-fold cross validation.

Summary

- Measuring Performance
 - Misclassification Rate & Accuracy
 - Precision & Recall
 - Balance accuracy measures
- ROC Analysis
- Overfitting
- Experimental Setup
 - Hold-out validation
 - k -Fold cross validation

References

- J. D. Kelleher, B. Mac Namee, A. D'Arcy. "Fundamentals of Machine Learning for Predictive Data Analytics", 2015.
- C. D. Manning, P. Raghavan and H. Schütze. "Introduction to Information Retrieval", Cambridge University Press. 2008
- E. Alpaydin. "Introduction to Machine Learning", Adaptive Computation and Machine Learning series, MIT press, 2009.
- J. Davis, M. Goadrich. "The relationship between Precision-Recall and ROC curves". Proceedings ICML 2006.