

Table of contents

1. Introduction

2. Model Evaluation and Selection (Text: 8.5)

- 2.1. Evaluation metrics for classification (Text 8.5.1)
- 2.2. Estimating a classifier's accuracy (Text: 8.5.2-8.5.4)
- 2.3. ROC Curve (Text: 8.5.6)
- 2.4. Comparing classifiers (Text: 8.5.5)
- 2.5. Exercises
- 2.6. Other issues affecting the quality of a model

3. Practical Exercises: Evaluation

1. Introduction

Most of this material is derived from the text, Han, Kamber and Pei, Chapter 8 and 9, or the corresponding powerpoint slides made available by the publisher. Where a source other than the text or its slides was used for the material, attribution is given. Unless otherwise stated, images are copyright of the publisher, Elsevier.

Here, we will discuss how to evaluate the performance of classifiers. When we have built a model using some learning algorithm or by fitting a statistical distribution, how do we know whether it is any good?

2. Model Evaluation and Selection (Text: 8.5)

Now that you may have built a classification model, there may be many questions going through your mind. For example, suppose you used data from previous sales (**training data**) to build a classifier to predict customer purchasing behaviour. You would like an estimate of how accurately the classifier can predict the purchasing behaviour of future customers (**test data**), that is, future customer data on which the classifier has not been trained. You may even have tried different methods to build more than one classifier and now you wish to compare their quality and choose the best one. For this, you will be most interested in the *accuracy* of the classifier. But

- What is accuracy?
- How can we estimate it?
- Are some measures of a classifier's accuracy more appropriate than others?
- How can we obtain a reliable accuracy estimate?

These questions are addressed in this section.

2.1. Evaluation metrics for classification (Text 8.5.1)

Accuracy, defined to be the proportion of correctly labelled tuples, is not the only measure to evaluate performance of classification. To understand the other measures, we first need to look at the **confusion matrix**.

Confusion Matrix (also called Error Matrix)

A confusion matrix is a useful tool for analysing how well a classifier can recognise tuples of different classes. Given a binary classification problem, a confusion matrix is a 2 by 2 matrix where each entry indicates the number of tuples categorised by the *actual* class (positive or negative label in training or testing data) vs *predicted* class (positive or negative predicted class suggested by the classifier).

Actual class (rows) \ Predicted class (columns)	C1=Positive	C2=Negative
C1=Positive	True Positives (TP)	False Negatives (FN)
C2=Negative	False Positives (FP)	True Negatives (TN)

From the confusion matrix, we can define four important measures:

- **True Positive** (TP): Number of *positive* tuples that were *correctly* labelled positive by the classifier
- **True Negative** (TN): Number of *negative* tuples that were *correctly* labelled negative by the classifier
- **False Positive** (FP): Number of *negative* tuples that were *incorrectly* labelled as positive by the classifier
- **False negative** (FN): A number of *positive* tuples that were *incorrectly* labelled as negative by the classifier

TP + TN is the number of tuples *correctly* labelled by the classifier (hence called *True*).

FP + FN is the number of tuples *incorrectly* labelled by the classifier (hence called *False*).

Note that all the FP tuples are *actually* N and all the FN tuples are *actually* P. That is, under this naming convention, the first character tells you if the classifier got it right (T) or wrong (F), and the second character tells you if the classifier predicts positive (P) or negative (N). The *actual* label for the tuple is not given in the name, but you can derive it.

Beware: There are several popular conventions for the layout of these matrices: sometimes postiveness is top and left as here but sometimes bottom and right; sometimes actuals are rows and predicted are columns, as here, but sometimes actuals are columns and predictions are rows. You need to pay attention to the table layout.

Example of confusion matrix (actuals are rows and predicted are columns)

Classes	<i>buys_computer</i> = yes	<i>buys_computer</i> = no	Total	Recognition (%)
<i>buys_computer</i> = yes	6954	46	7000	99.34
<i>buys_computer</i> = no	412	2588	3000	86.27
Total	7366	2634	10,000	95.42

- Confusion matrix for the classes *buys_computer* = yes and *buys_computer* = no
- For example
 - 6954: The number of positive tuples classified as positive -- TP
 - 412: The number of negative tuples classified as positive -- FP
 - 46: The number of positive tuples classified as negative -- FN
 - 2588: The number of negative tuples classified as negative -- TN

Various evaluation measures from a confusion matrix

Let

- P = the number of tuples actually positive in the training data
- N = the number of tuples actually negative in the training data

With four primitive measures, we can define some important evaluation measures as follows:

<i>Measure</i>	<i>Formula</i>
accuracy, recognition rate	$\frac{TP + TN}{P + N}$
error rate, misclassification rate	$\frac{FP + FN}{P + N}$
sensitivity, true positive rate, recall	$\frac{TP}{P}$
specificity, true negative rate	$\frac{TN}{N}$
precision	$\frac{TP}{TP + FP}$
F , F_1 , F -score, harmonic mean of precision and recall	$\frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$
F_β , where β is a non-negative real number	$\frac{(1 + \beta^2) \times \text{precision} \times \text{recall}}{\beta^2 \times \text{precision} + \text{recall}}$

- This table shows some basic evaluation measures for classifications.
- accuracy = 1 - error rate

Class Imbalance Problem: Beware

One may wonder why we need such a range of evaluation measures. At first, the accuracy seems to be enough for a classification task, but the accuracy may not be a good way to show the performance of your classifier when the dataset is *unbalanced*.

An **unbalanced dataset is one where the classes are not evenly distributed in the data**, ie far from 50% each in the binary classification case. One class dominates the data; usually the positive class is rare.

Consider the following example that shows a confusion matrix for a cancer classification. Again, actuals are rows and predicted are columns in the table.

<i>Classes</i>	<i>yes</i>	<i>no</i>	<i>Total</i>	<i>Recognition (%)</i>
<i>yes</i>	90	210	300	30.00
<i>no</i>	140	9560	9700	98.56
Total	230	9770	10,000	96.40

If we only care about the classifier's accuracy, then 96.4% appears to be a good result at first glance, quite close to 100%.

Wrong!

Let's consider a classifier that we have learnt which classifies every patient as "cancer=no". Clearly we did not need a complex data mining algorithm to learn this ridiculously simple classifier, a majority vote. In this case, we have 97.7% accuracy (9770/10000). Not bad, huh? **Wrong!** Our new classifier is telling us nothing, only the distribution of the classes. And note our first classifier above performed even worse than this according to accuracy, so it was unacceptably poor.

Clearly, an accuracy rate of 97% is not acceptable on this problem—a classifier with this accuracy could be correctly labelling only the noncancer tuples and misclassifying all the cancer tuples as our "cancer=no" classifier does. Instead, we need other measures, which can distinguish how well the classifier can recognise the positive tuples (cancer = yes) and how well it can recognise the negative tuples (cancer = no).

The **sensitivity** and **specificity** measures can be used, respectively, for this purpose.

For example, the sensitivity and specificity of the above example are:

- $sensitivity = \frac{TP}{P} = \frac{90}{300} = 0.30$
- $specificity = \frac{TN}{N} = \frac{9560}{9700} = 0.99$

Thus, we note that although the classifier has a high accuracy, it's ability to correctly label the positive (rare) class is poor as given by its low sensitivity. It has high specificity, meaning that it can recognise negative tuples quite well. The sensitivity is much more important than specificity in this case, due to the purpose of the classification task.

But what if our classifier could deliver 100% sensitivity? Too easy: the classifier "cancer=yes" can do this. **Is this a good result?** **No!** Accuracy would be only 3%. Specificity would be 0%. Predicting all people have cancer is just as useless in practice as predicting no people have cancer.

Sensitivity and specificity are typically a **tradeoff**, you can maximise one by reducing the other: 100% for each is ideal (well, maybe not, due to potential *overfitting* discussed later), but the tradeoff between them, and whether some classifier is therefore *good enough*, is something for the expert to interpret with knowledge of the underlying purpose.

The **precision** and **recall** measures, originally developed for information retrieval, are also widely used in classification as alternative tradeoff quality measures:

- Precision can be thought of as a measure of exactness
 - i.e., what percentage of tuples classified as positive are actually such
- Recall (equivalent to *sensitivity*, above) is a measure of completeness
 - i.e., what percentage of positive tuples are classified as such

Often precision and recall are combined into an *F1-score*, which is the harmonic mean of the precision and recall. It might also be called simply *f-score*, or *f-measure*. See the table above for its formulation. Maximising f-score is useful because it provides a single measure, and takes account of potentially unbalanced data by focusing on the positive class, but it emphasises a particular relationship between right and wrong predictions. **Is this the right quality measure for your classification task? Or not?**

ACTION: Calculate precision, recall and f-measure for the cancer classification confusion matrix above as well as for the classifiers "cancer=yes" (which classifies every tuple as positive) and "cancer=no" (which classifies every tuple as negative). Comment on the interpretation of f-measure for this problem. A worked solution is here: [!\[\]\(104fbf564e2e5a8fbd84f31656d114c7_img.jpg\)Solution to Exercise: Evaluation measures](#)

Key message: Always consider whether your measure of performance is appropriate for your problem. Choose an appropriate measure which might be influenced by the practice in the domain of application. Always consider how your performance compares to a dumb classifier which might be 50% accuracy for a balanced dataset but something else for unbalanced data.

2.2. Estimating a classifier's accuracy (Text: 8.5.2-8.5.4)

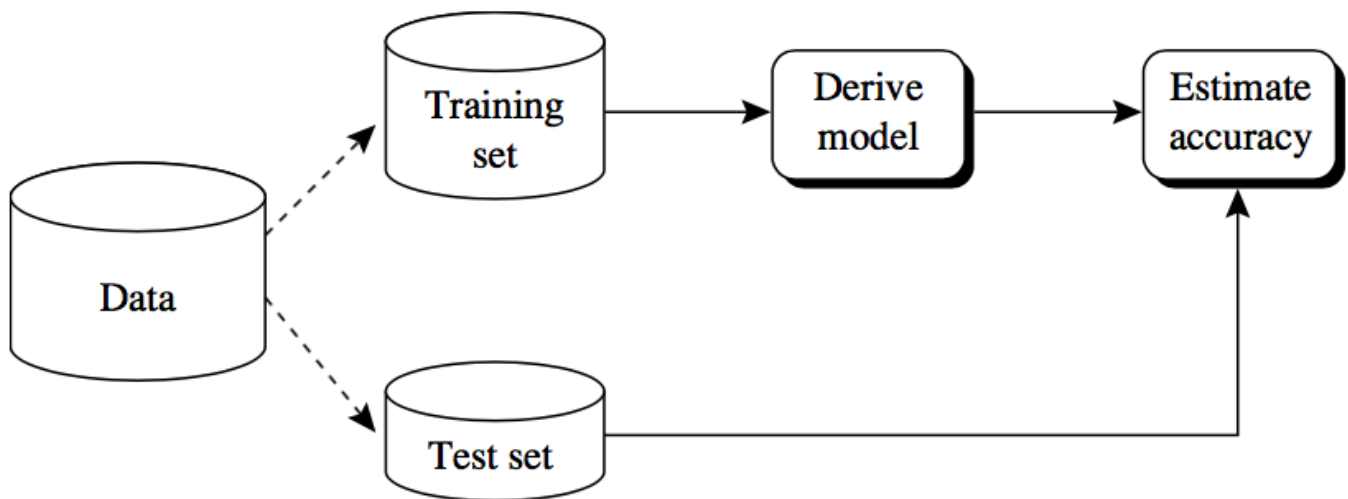
In this section, we will see how to report the performance of the models by various methods.

Accuracy (or error rate) on training data is not a good indicator of future performance because the model may be overly-tuned towards exactly the data on which it was trained. For example, consider the model which internally simply remembers the data it has seen and classifies that data as it was classified in the training data, and every new data item is classified arbitrarily. Would you expect this model to work well?

This situation is called **overfitting** and commonly leads to poor performance on unseen data.

Holdout method

- Given data is randomly partitioned into two independent sets
 - Training set (e.g., 2/3) for model construction
 - Test set (e.g., 1/3) for accuracy estimation
 - Overall flow of holdout method:



Estimating accuracy with the holdout method.

Training / Validation / Test

Variation of holdout method when you want to experiment with parameters of the model-building algorithm.

- Randomly partition the given data into three different sets:
- Training set
 - To construct (i.e *train*) a classification model
- Validation set
 - To find the best parameters for the model, likely to be done by a person studying the effect of various parameters on the accuracy (or other quality measures) of the training set.
- Test set
 - To measure the final performance of the model as it would be reported.

Cross-validation

- k-fold, where $k = 10$ is most popular (due to low bias and variance)
- Randomly partition the data into k mutually exclusive subsets D_1, D_2, \dots, D_k , each approximately equal size
- At i -th iteration, use D_i as test set and others as training set
- **Leave-one-out**: Special case for *small-sized datasets*: Use k folds where $k = \text{number of tuples}$,
- **Stratified cross-validation**: Special case where folds are not randomly selected but stratified so that the class distribution in each fold is approximately the same as that in the initial data (to achieve low bias).
- Overall accuracy is computed as the average accuracy of each model on its respective test set.

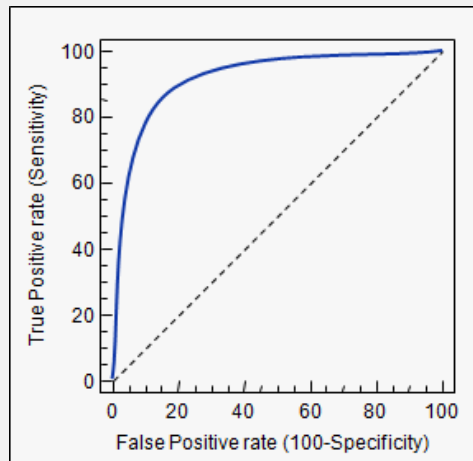
Bootstrap

- Works well for small data sets where, otherwise, the requirement to split the data into training and testing sets makes both sets too small for purpose.
- Samples the given training tuples uniformly **with replacement**
 - i.e., each time a tuple is selected, it is equally likely to be selected again and added to the training set again.
- Several bootstrap methods : a common one is **.632 bootstrap**
 - A data set with d tuples is sampled d times, with replacement, resulting in a training set of d samples. The data tuples that did not make it into the training set end up forming the test set. About 63.2% of the original data end up in the training set, and the remaining 36.8% form the test set (since $(1 - 1/d)d \approx e^{-1} = 0.368$ if d is very large.)
 - Repeat the sampling procedure k times, overall accuracy of the model is:

$$Acc(M) = \frac{1}{k} \sum_{i=1}^k (0.632 \times Acc(M_i)_{test_set} + 0.368 \times Acc(M_i)_{train_set}),$$

where $Acc(M_i)_{testset}$ is the accuracy of the model trained with training set i when it is applied to test set i and $Acc(M_i)_{trainset}$ is the accuracy of the model obtained with training set i when it is applied to the training set i .

2.3. ROC Curve (Text: 8.5.6)



Example of ROC curve (axes are labelled with percentages in this diagram).
The dashed diagonal line indicates when $TPR = FPR$ and has an area underneath it of 0.5

ROC (Receiver Operating Characteristics) curves: for visual comparison of probabilistic classification models and selection of a decision threshold. Also for visually comparing tradeoffs in performance for alternative deterministic classifiers.

- **Basic idea:** a *probabilistic classifier* returns a probability of a tuple being in the positive class. What if we consider a probability threshold for positive classification being somewhere in the range $[0, 1]$ instead of using the simple 0.5 (majority vote for the class)? This is a way to recognise that the cost of errors (ie FP vs FN) may not be equal for each class.
- Shows the **trade-off** between the true positive rate (TPR) and the false positive rate (FPR)
 - TPR (= sensitivity) is the proportion of positive tuples that are correctly labelled by the model: TP/P
 - FPR (= 1- specificity) is the proportion of negative tuples that are mislabelled as positive: FP/N
 - A deterministic classifier (which assigns classes without probabilities) can be plotted as a single point on the ROC chart (the point is (FPR, TPR)).
 - A probabilistic classifier is plotted as a ROC curve on the chart (see below).
- Use the ROC curve to choose a decision threshold for your probabilistic classifier that reflects the tradeoff you need, ideally the probability corresponding to an inflexion point where the curve turns from vertical to horizontal, so that you are getting the benefit of near-maximal TPs with near-minimal FPs. Selection and use of the decision threshold at this point turns your probabilistic classifier into a deterministic one plotted at that point.
- The area under a ROC curve (**ROC-AUC**) is often used to measure the performance of a *probabilistic* model.
- The area under a ROC curve (**ROC-AUC**) can also be computed for a *deterministic* model as the area under the curve constructed by drawing a line from $(0,0)$ to (FPR, TPR) and another from (FPR, TPR) to $(1,1)$. By geometric analysis, it is easy to see that this equates to the average of sensitivity and specificity, i.e. $(TP/P + TN/N) / 2$.
- The diagonal line on the graph represents a model that randomly labels the tuples according to the distribution of labels in the data. This line has AUC of 0.5. A model better than random should appear *above* the diagonal. The closer a model is to random (i.e., the closer its ROC-AUC is to 0.5), the poorer is the model. A model falling below the diagonal line is worse than random (which is very, very poor, but hopefully you are building better models than that!).
- Many deterministic models with distinct (FPR, TPR) points on the graph share the same ROC-AUC, falling on an isometric line parallel to the $AUC=0.5$ diagonal. While these models have different performance on P and N examples, ROC-AUC alone does not distinguish them. A visual study of the chart might be helpful.
- For a probabilistic model, the ROC curve may cross the diagonal line for some probabilities; but it may still be a good model if the ROC-AUC is high.

- A ROC-AUC of 1 indicates a perfect classifier for which all the actual P tuples have a higher probability of being labelled P than all the actual N tuples. A ROC-AUC of 0 is the reverse situation: all the actual Ps are less likely to be labelled P than all the actual Ns, denoting a worst case model.
- The ROC-AUC represents the proportion of randomly drawn pairs (one from each of the two classes) for which the model correctly classifies both tuples in the random pair. In contrast to accuracy or error rate, ROC-AUC allows for unbalanced datasets by counting the performance over the subsets T (on the y axis) and N (on the x axis) independently, valuing errors in each class of the dataset independently of the proportion of each class in the dataset as a whole.
- Instead of probabilities generated by a probabilistic classifier, the ROC can also be used to choose a cost or risk function to be used with a deterministic classifier.
- The ROC may be used together with cross-validation so it is not overly influenced by a particular training set.

Plotting ROC curve for a probabilistic classifier

- ROC curve can be plotted with a probabilistic classifier (e.g. naive Bayes, some decision trees, neural nets)
- The vertical y axis of an ROC curve represents TPR. The horizontal x axis represents FPR.
 1. **Rank the test tuples in decreasing order:** the one that is most likely to belong to the positive class (highest probability) appears at the top of the list.
 2. Starting at the bottom left corner (where $TPR = FPR = 0$), we check the tuple's actual class label at the top of the list. If we have a **true positive** (i.e., a positive tuple that was correctly classified), then true positive (TP) and thus TPR increase.
 - On the graph, we **move up and plot a point**.
 3. If, instead, the model classifies a **negative tuple as positive**, we have a false positive (FP), and so both FP and FPR increase.
 - On the graph, we **move right and plot a point**.
 4. This process is repeated for each of the test tuples in ranked order, each time moving up on the graph for a true positive or toward the right for a false positive.

Example 1

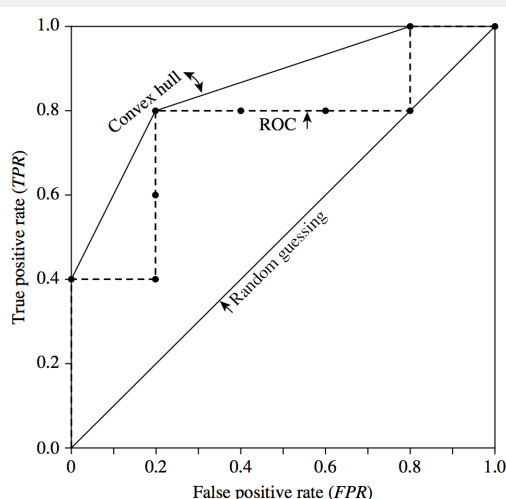
The following table shows the probability value of being in the positive class that is returned by a probabilistic classifier (column 3), for each of the 10 tuples in a test set. Column 2 is the actual class label of the tuple. There are five positive tuples and five negative tuples, thus $P = 5$ and $N = 5$. As we examine the known class label of each tuple, we can determine the values of the remaining columns, TP, FP, TN, FN, TPR, and FPR.

<i>Tuple #</i>	<i>Class</i>	<i>Prob.</i>	<i>TP</i>	<i>FP</i>	<i>TN</i>	<i>FN</i>	<i>TPR</i>	<i>FPR</i>
1	<i>P</i>	0.90	1	0	5	4	0.2	0
2	<i>P</i>	0.80	2	0	5	3	0.4	0
3	<i>N</i>	0.70	2	1	4	3	0.4	0.2
4	<i>P</i>	0.60	3	1	4	2	0.6	0.2
5	<i>P</i>	0.55	4	1	4	1	0.8	0.2
6	<i>N</i>	0.54	4	2	3	1	0.8	0.4
7	<i>N</i>	0.53	4	3	2	1	0.8	0.6
8	<i>N</i>	0.51	4	4	1	1	0.8	0.8
9	<i>P</i>	0.50	5	4	0	1	1.0	0.8
10	<i>N</i>	0.40	5	5	0	0	1.0	1.0

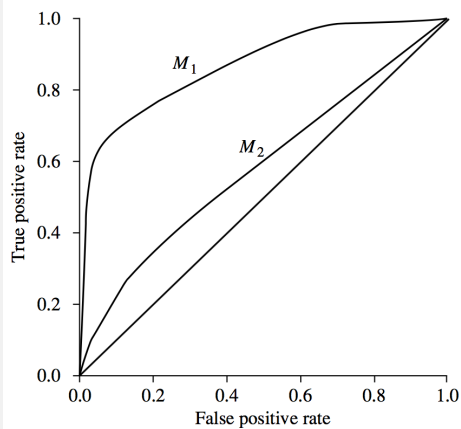
Tuples sorted by decreasing score, where the score is the value returned by a probabilistic classifier.

We start with tuple 1, which has the highest probability score and take that score as our threshold, that is, $t = 0.9$. Thus, the classifier considers tuple 1 to be positive, and all the other tuples are considered negative. Since the actual class label of tuple 1 is positive, we have a true positive, hence $TP = 1$ and $FP = 0$. Among the remaining nine tuples, which are all classified as negative, five actually are negative (thus, $TN = 5$). The remaining four are all actually positive, thus, $FN = 4$. We can therefore compute $TPR = TP = 1 = 0.2$, while $FPR = 0$. Thus, we have the point $(0.2, 0)$ for the ROC curve.

Next, threshold t is set to 0.8, the probability value for tuple 2, so this tuple is now also considered positive, while tuples 3 through 10 are considered negative. The actual class label of tuple 2 is positive, thus now $TP = 2$. The rest of the row can easily be computed, resulting in the point $(0.4, 0)$. Next, we examine the class label of tuple 3 and let t be 0.7, the probability value returned by the classifier for that tuple. Thus, tuple 3 is considered positive, yet its actual label is negative, and so it is a false positive. Thus, TP stays the same and FP increments so that $FP = 1$. The rest of the values in the row can also be easily computed, yielding the point $(0.4, 0.2)$. The resulting ROC graph, from examining each tuple, is the jagged line as follows. A convex hull curve is then fitted to the jagged line as shown.



Example 2



ROC curves of two probabilistic classification models, M_1 and M_2 . The diagonal shows where, for every true positive, we are equally likely to encounter a false positive. The closer a ROC curve is to the diagonal line, the less accurate the model is. Thus M_1 is more accurate here. If the ROC curves for M_1 and M_2 cross over then varying the threshold selection will vary which is more accurate for binary classification.

2.4. Comparing classifiers (Text: 8.5.5)

Classifier Models M1 vs. M2

- Suppose we have 2 classifiers, M1 and M2. Which one is better?
- Use 10-fold cross-validation to obtain mean error rates for M1 and M2, $err(M1), err(M2)$
- It may seem intuitive to choose the model with the lowest error rate
- **However**, these mean error rates are just **estimates** of error on the true population of future data cases
- What if the difference between the 2 error rates is just attributed to chance?
 - Use a test of statistical significance
 - Obtain confidence limits for our error estimates


Estimating Confidence Intervals: Null Hypothesis

- **Null Hypothesis: M1 & M2 are the same**
- Test the null hypothesis with **t-test**
 - Use t-distribution with k-1 degree of freedom
- If we can reject null hypothesis, then
 - we conclude that the difference between M1 & M2 is statistically significant.
 - Chose model with lower error rate
- Perform 10-fold cross-validation (k=10)
 - For i -th round of 10-fold cross-validation, the same cross partitioning is used to obtain $err(M1)_i$ and $err(M2)_i$.
 - Average over 10 rounds to get $\overline{err}(M1) = \frac{1}{k} \sum_{i=1}^k err(M1)_i$ and similarly for $\overline{err}(M2)$
 - t-test computes **t-statistic** with k-1 degrees of freedom:

$$t = \frac{\overline{err}(M1) - \overline{err}(M2)}{\sqrt{var(M1 - M2)/k}},$$

where (using the variance for the population, as given in the text):

$$var(M1 - M2) = \frac{1}{k} \sum_{i=1}^k [err(M1)_i - err(M2)_i - (\overline{err}(M1) - \overline{err}(M2))]^2.$$

- To determine whether M1 and M2 are significantly different, we compute t-statistic and select a significance level.
 - 5% significance levels: The difference between M1 and M2 is significantly different for 95% of population.
 - 1% significance levels: The difference between M1 and M2 is significantly different for 99% of population.
- Based on t-statistics and significance level, we consult a table for the t-distribution.
 - We need to find the t-distribution value corresponding to k - 1 degrees of freedom (or 9 degrees of freedom for our example) from the table (Two-sided).
 -  [T-distribution table](#)
- If the t-statistic we calculated above is *not* between the corresponding value in the table and its negative (i.e. the corresponding value in the table multiplied by -1), then we reject the null hypothesis and conclude that M1 and M2 are significantly different (at the significance level we chose above).
- Alternatively, if the t-statistic we calculated above *is between* the corresponding value in the table and its negative, we conclude that M1 and M2 are essentially the same and any difference is attributed to chance.

2.5. Exercises

ACTION: Try out this exercise.

 [Exercise: Model Selection using t-test](#)

When you have had a go, you can check your answers against this worked answer:

 [Solution to Exercise: Model Selection using t-test](#)

2.6. Other issues affecting the quality of a model

A model may be preferred over another for a number of reasons. These reasons will influence the choice of a learning method as well as the selection of a particular classifier produced by the method.

- Accuracy
 - Classifier accuracy: predicting class label
- Speed
 - Time to construct the model (training time)
 - Time to use the model (classification/prediction time)
- Robustness: handling noise and missing values
- Scalability: efficiency in disk-resident databases
- Interpretability
 - Understanding and insight provided by the model. This is especially important in cases where the model is never intended to be put into practice over unseen data, but instead to influence systemic behaviours such as business rules and policies. It may also be critical to enable qualitative evaluation of the model for embedded bias.
- Availability and Trust
 - Does the business environment have the technical infrastructure, skills and policy or governance framework to use it?
 - Will the business environment trust the results to be used for the intended purpose?
- Other measures specific to the method, e.g., goodness of rules, such as decision tree size or compactness of classification rules.


Note well that these kind of factors are just as influential on the selection of a method and a model for mining problems other than classification and prediction. For example, association rules are great for interpretability and scalability, but may not be considered trustworthy.

3. Practical Exercises: Evaluation

ACTION: Attempt these practical exercises with Rattle. There is a video showing the mechanics to get you started, written instructions for you to work through, and separately some suggested solutions.

COMP3425/8410 Evaluation Methods in Rattle



 [Practical Exercise: Evaluation](#)

 [Solution to Practical Exercises: Evaluation](#)