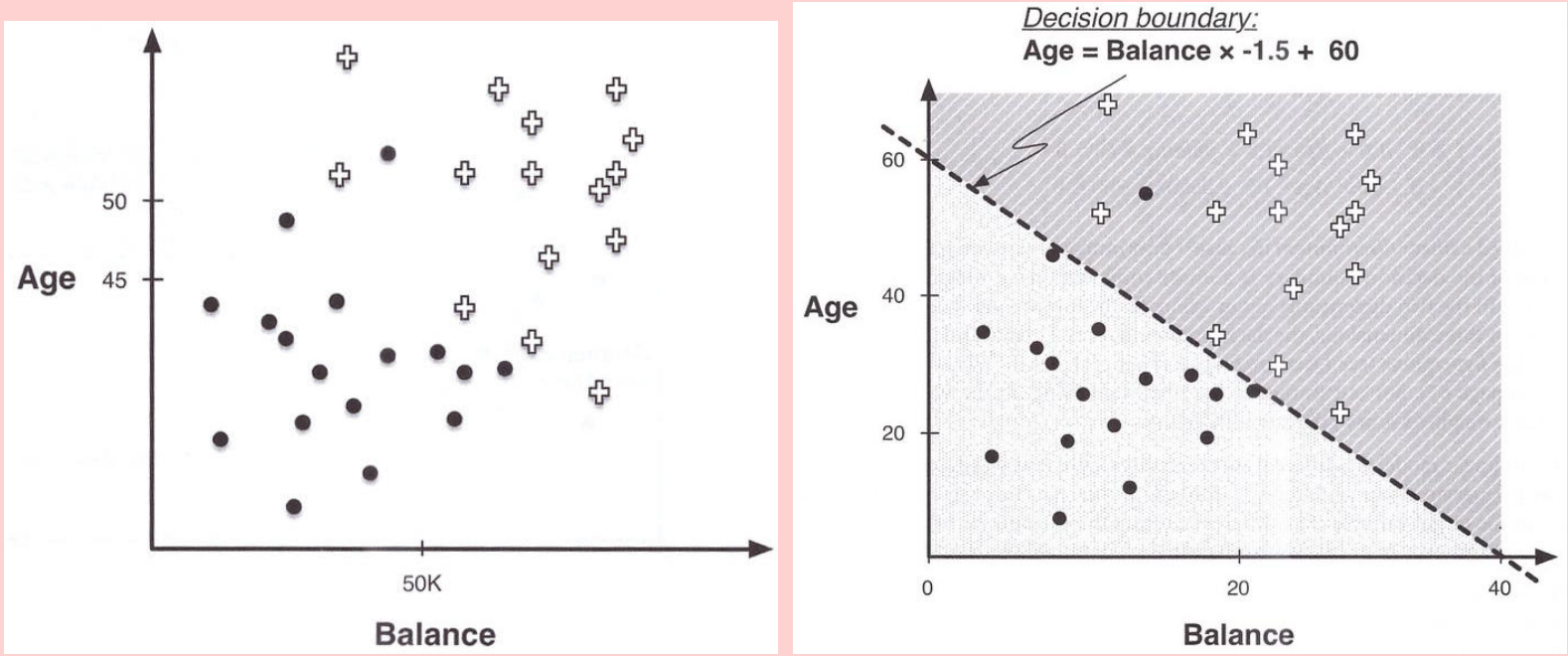
PSI:
Logistic Regression

Technological University Dublin City Campus

School of Computer Science

Linear classifiers

- A linear classifier has numeric inputs and a categorical output.
- In a model with two predictor variables (inputs) and a dichotomous dependent variable (output), a linear classifier can be represented geometrically as a straight line that splits the entire space into two parts in an attempt to separate the points belonging to different classes (i.e. having different values of the categorical variable).
- For a higher number of independent variables, the dividing artefact is an $(n-1)$-dimensional plane, where n is the number of independent variables.



Source: [DSB]

In this example, the categorical dependent (predicted) variable indicates if a bank customer will default on a loan. The predictor variables are ***Balance*** *and* ***Age****.*

- The form of the general linear model is the same as for regression:

$$f(x) = w_0 + w_1 x_1 + w_2 x_2 ... + w_n x_n$$

  where n is the number of variables, x_i are the independent variables and w_i are weights. The function $f(x)$ is called the *linear discriminant function*.

- The fitting of a linear classifier model is a process of optimisation, where an objective (more on next page) is pursued for $f(x) = 0$ (this equation defines an $(n-1)$-dimensional plane, where n is the number of predictor variables)

- Once model is fitted the linear discriminant function can be used to assign new data instances to classes, with $f(x) > 0$ placing data instance x in one class and $f(x) \leq 0$ in the other.

- If the values of the predictors are normalised before fitting, the weights, w_i, can be interpreted as a measure of their relative importance.

- Different linear classifiers **differ by their optimisation objective**.
- Examples of optimisation objectives:
  - logistic and probit regression use maximum likelihood estimation (MLE), which in the case of these models is based around maximising a certain measure of distance from the discriminant, summed over the data sample
  - support vector machines (SVM) maximise a strip of 'no man's land' around the discriminant, with penalties for 'intruding' points

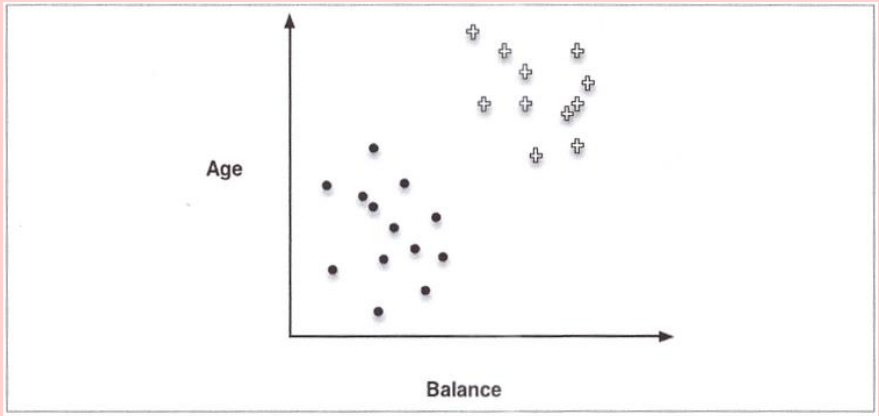


Figure 4-4. A basic instance space in two dimensions containing points of two classes.

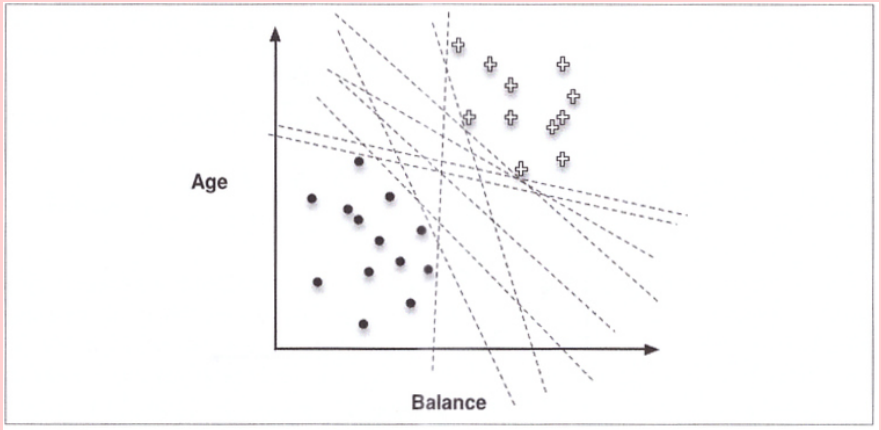
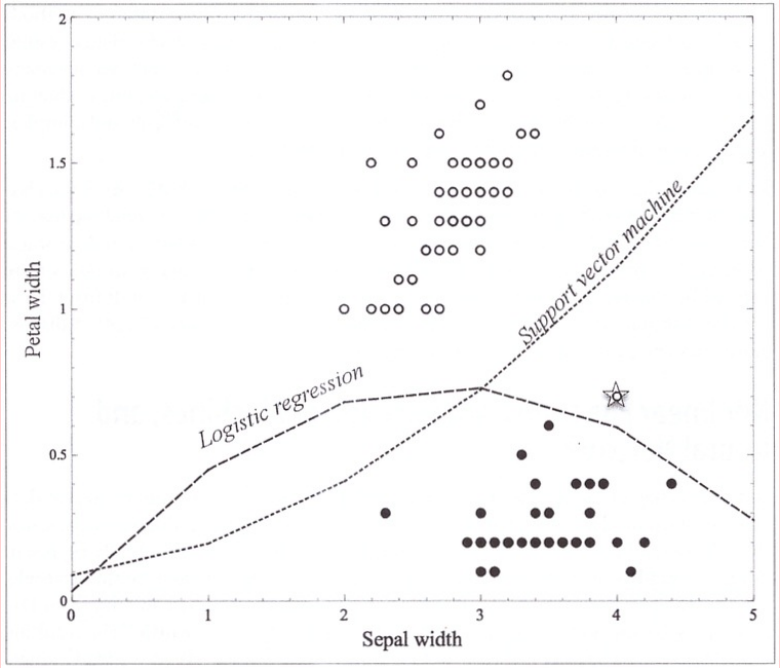


Figure 4-5. Many different possible linear boundaries can separate the two groups of points of Figure 4-4.

Source: [DSB]

Non-linear classifiers

Classification methods may use non-linear functions (where the attribute values contribute with a power other than 1). In this case the boundary line for classification is not straight.



The picture shows data being classified using non-linear logistic regression and a non-linear support vector machine.

Source: [DSB]

Logistic regression

- Logistic regression is in fact, despite its name, a classification model. Its input attibutes are numeric and the output is categorical.

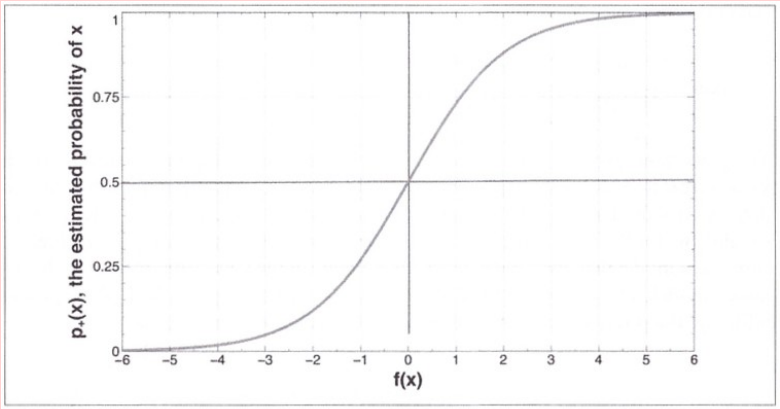


Figure 4-10. Logistic regression's estimate of class probability as a function of f(x), (i.e., the distance from the separating boundary). This curve is called a "sigmoid" curve because of its "S" shape, which squeezes the probabilities into their correct range (between zero and one).

Source: [DSB]

- The model is fitted using the following sigmoid function, which is a transformation of the linear discriminant: $\boxed{p_+(\mathbf{x}) = \frac{1}{1 + e^{-f(\mathbf{x})}}}$
- The value of $p_+(\mathbf{x})$ is in the range [0,1] but is not a probability. However, once the model is fitted, calculating the sigmoid function on unseen data can easily be used for classification (over 0.5 results in classification as '+' and below 0.5 as 'not +').

- If we treat $p_+(\mathbf{x})$ as a probability, then the (discrete) probability distribution function of the response (dependent) variable y with values 1 and 0 is: $(p_+(x))^y(1-p_+(x))^{(1-y)}$ and the probability of the whole sample response, given the sample predictor values and a particular $f(x)$, is:

$$\prod_{i=1}^{n}(p_+(\mathbf{x}))^y(1-p_+(\mathbf{x}))^{(1-y)}$$

- The product derived above is the likelihood of the observed data and logistic regression models are fitted through its maximisation using an iterative algorithm for finding the best weights w_i (there isn't a closed-form expression like for linear regression).

- The form actually maximised is the logarithm of the product, corresponding to the **log-likelihood**:

$$\sum_{i=1}^{n} ylog(p_+(\mathbf{x})) + (1-y)log(1-p_+(\mathbf{x}))$$

Logistic regression assumptions

- **Linearity** of relationship between the logit (log-odds) function and any numeric predictor (Box-Tidwell test can be performed to check this)
- **Independence of errors** says that the instances (observations) must not be related. This must be part of the analysis design.
- **Absence of multicollinearity** can be checked via the variance inflation factor (VIF) values for the independent variables.

Other similar models

- **Ordinal logistic regression** is used to predict an ordinal variable (e.g. the answers to a Likert-scale question) - this is based on the same idea as dichotomous logistic regression
- **Probit regression** - this is very similar to logistic regression, but uses the cumulative form of the normal distribution for range conversion, in contrast to the log-odds used by logistic regression

Evaluating the model

- The **deviance** or **-LL** is calculated from the log-likelihood: $\boxed{deviance = -2LL}$

  It has the χ^2 distribution and smaller values are associated with better models. The difference in $-2LL$ between two models can be used as a significance indicator of improvement.

- The Akaike information criterion (**AIC**) and Bayes information criterion (**BIC**) are calculated as $\boxed{AIC = -2LL + 2k}$ and $\boxed{BIC = -2LL + 2k \times log(n)}$

  These measures include a marginal penalty for each predictor added to the model, to compensate for the overfitting introduced by adding predictor variables. The are used for comparison of models.

- Significance of the coefficients of the fitted model is investigated through the z statistic: $\boxed{z = \frac{b}{SE_b}}$

- The odds ratio for a change of one unit in the value of predictor x_i gives a measure of the effect size of the corresponding coefficient: $\boxed{OR = e^{\beta_i}}$

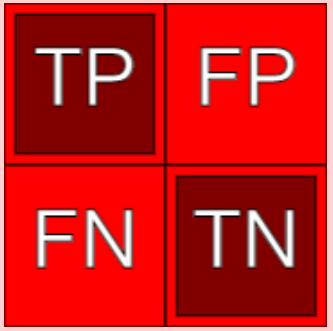
Confusion matrix for evaluation of classification

- **true positive (TP)**: the number of items correctly identified by the model as belonging to the 'positive' class
- **false positive (FP)**: the number of items belonging to the 'negative' class incorrectly identified by the model as belonging to the 'positive' class
- **false negative (FN)**: the number of items belonging to the 'positive' class incorrectly identified by the model as belonging to the 'negative' class
- **true negative (TN)**: the number of items correctly identified by the model as belonging to the 'negative' class

For categorical variables with **more than two values**, the confusion matrix has more than two columns and rows.

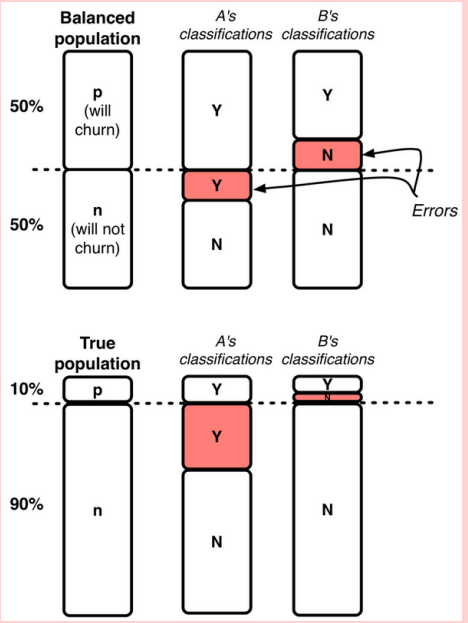
- **Accuracy**
  - most commonly used measure for quick evaluation
  - measures the proportion of predictions that are correct
  - a 'blunt instrument'
  - but easily calculable:

$$accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$



  - depends on the mix of the predicted classes

    In the picture model A predicts positives perfectly but predicts negatives with 60% accuracy. B predicts negatives perfectly but predicts positives with 60% accuracy. The result is that when the class mix is 50-50 the overall accuracy is the same for the two models but when the mix is 10-90 model B comes out as a lot better.



- The **rates**:

true positive rate: $\frac{TP}{TP+FN}$

false negative rate: $\frac{FN}{TP+FN}$

true negative rate: $\frac{TN}{FP+TN}$

false positive rate: $\frac{FP}{FP+TN}$

- statistics measures

  sensitivity (equal to the *true positive rate*): $\frac{TP}{TP+FN}$

  specificity (equal to the *true negative rate*): $\frac{TN}{FP+TN}$

[Confusion matrix for evaluation of classification]

- text classification and information retrieval measures

  precision (or **positive predictive value**): $\frac{TP}{TP+FP}$

  recall (equal to the *true positive rate*): $\frac{TP}{TP+FN}$

  F-measure (harmonic mean of precision and recall):

$$F\text{-}measure = 2 \times \frac{precision \times recall}{precision + recall}$$

Cost and benefit in classification evaluation

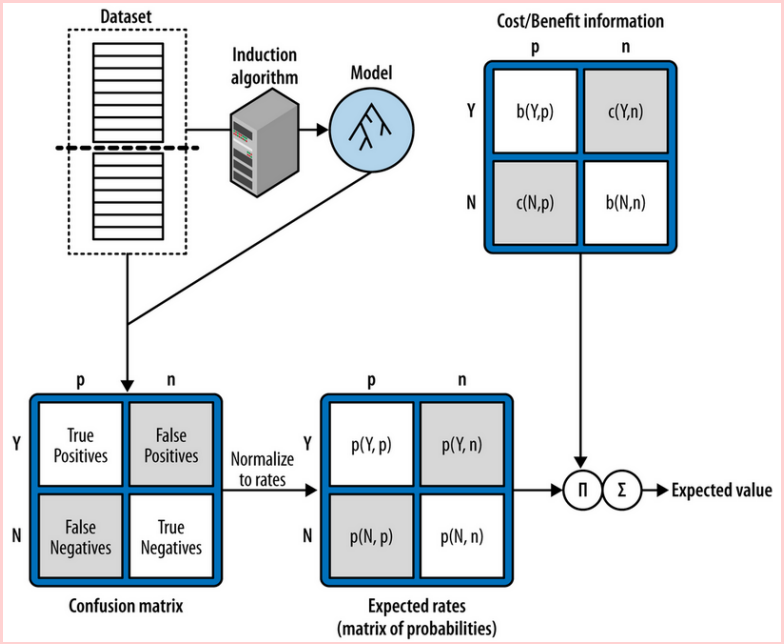
- **Expected value** in general is a value given to a situation that has a countable number of known outcomes with known probabilities. If the value of each outcome is also known or can be estimated, then the expected value is:

$$EV = p(o_1)v(o_1) + p(o_2)v(o_2) + ... + p(o_n)v(o_n)$$

  where o_1, o_2, ... o_n are the outcomes, n is the number of outcomes, $p(o_i)$ is the probability of outcome o_i and $v(o_i)$ is the value of outcome o_i.

- In a data analytical context the values of $p(o_i)$ and $v(o_i)$ are determined as follows:
  - the **probabilities** are estimated from the data, by building a model
  - the **value** of each outcome must be determined using:
    - general knowledge of the problem domain
    - particular knowledge of the specific problem at hand

- *Expected value* can be calculated using the confusion matrix



The confusion matrix is *normalised* i.e. the counts are converted to probabilities for the different outcomes. A matrix of the same dimensions is 'filled in' with cost and benefit values. The cell-for-cell multiplication of the matrices and summation of the values in the cells of the resulting matrix yields the *expected value*. Beware of: **duplication of values** and **cost/benefit sign mix-ups**.

$$expected\_value = p(\mathtt{Y},\mathtt{p})b(\mathtt{Y},\mathtt{p}) + p(\mathtt{N},\mathtt{p})c(\mathtt{N},\mathtt{p}) + p(\mathtt{N},\mathtt{n})b(\mathtt{N},\mathtt{n}) + p(\mathtt{Y},\mathtt{n})c(\mathtt{Y},\mathtt{n})$$

- Another form of the same equation has the a priori probabilities of `p` and `n` separated out, for easy calculation for data sets with different class mixes

$$\begin{aligned} expected\_value = & p(\mathtt{p})[p(\mathtt{Y}|\mathtt{p})b(\mathtt{Y},\mathtt{p}) + p(\mathtt{N}|\mathtt{p})c(\mathtt{N},\mathtt{p})] \\ & + p(\mathtt{n})[p(\mathtt{N}|\mathtt{n})b(\mathtt{N},\mathtt{n}) + p(\mathtt{Y}|\mathtt{n})c(\mathtt{Y},\mathtt{n})] \end{aligned}$$

Example

Expected value can be used to determine whether some action would be viable and in what cases. In **targeted marketing** the action is the sending of marketing material to a particular person or class of person. When working with a predictive model that outputs the probability of a person or class of person responding to marketing, making a targeting decision based on the intuitive threshold of 0.5 would not be very useful, as all the probabilities would be a lot lower than that threshold. In such cases, it is a lot more useful to look at the expected value (i.e. benefit) of targeting rather than use a simple threshold.

$$expected\_value\_of\_targeting = p_R(x)v_R + [1 - p_R(x)]v_{NR}$$

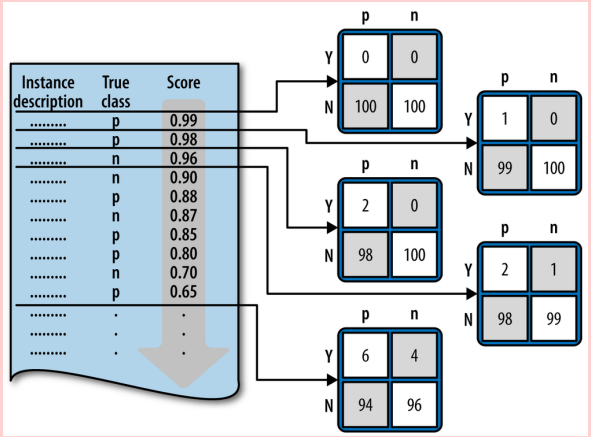
In the formula above, $p_R(x)$ is the probability of x (a person or a class of person) responding, v_R is the benefit of a response and v_{NR} is the value (which will be negative) of a non-response. Let's say that the targeted marketing is for a product that would bring a profit of €100 if the targeted person bought it (responding to the marketing action) and that the price of creating and sending the material is €1 per address. Then we would have:

$$v_R = €100 - €1 = €99, \qquad v_{NR} = -€1$$

For the values above, the expected value of targeting is greater than 0 when $p_R > 0.01$. Thus we may decide to target any person for whom this condition is met, expecting that such action would, on average, be of benefit.

Graphs for classification evaluation

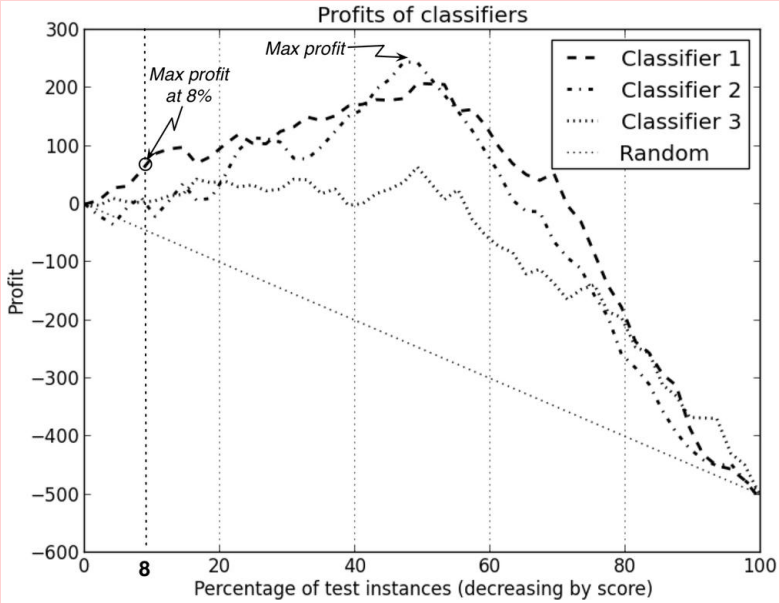
- Ranking of items - this is something that can be achieved with most classification models, whether they produce probabilities (e.g. Naïve Bayes) or values between 0 and 1 that do not actually correspond to probabilities (logistic regression). The data instances are simply ranked from the 'most probable' to the 'least probable' based on the model output.



- Why rank instead of classify? Because the standard decision boundary of 0.5 would not make sense for many applications. E.g. the earlier described direct marketing example and the boundary that would make sense is **not known in advance**.

- Ranking allows the decision boundary to be determined depending on profit and budget

Profit curves



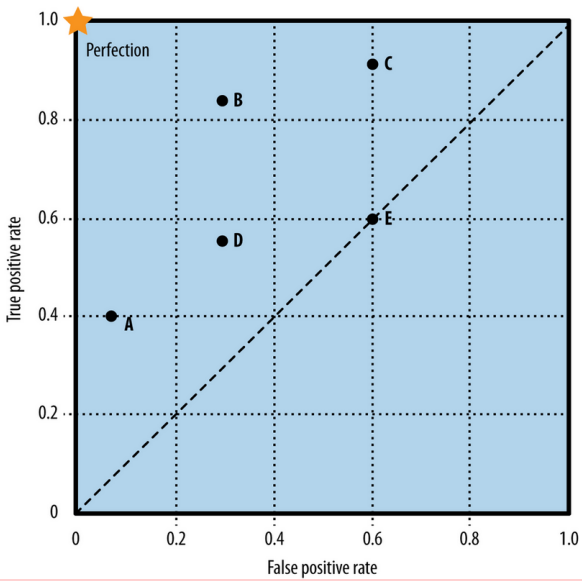
Source: [DSB]

Profit curves provide information about the performance of classifiers and can help with determining the best binary decision boundary for a data set and given classification methods. A profit curve can be used when the **prior probabilities of the classes** and the **cost-benefit matrix** are stable across all data sets (those used to build the prediction model and those to be classified). Examples of questions that a profit curve would help answer are "Which one of three classifiers is the best one to use if the budget is limited (e.g. to target-marketing 8% of unknown subjects)?" [Classifier 1 in the picture] or it might be "Which one of the three classifiers, when used to rank the new subjects, would bring about the hightest profits with an unlimited budget?" [Classifier 2 in the picture]

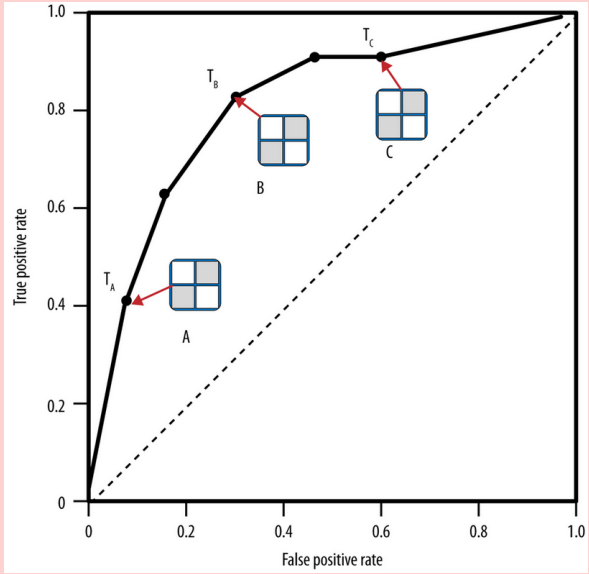
The x-axis of a profit curve represents the percentage of data instances that is found above the decision boundary as the boundary is moved, instance by instance, down the ranked (e.g. by probability to respond to targeted marketing) list of instances in a training set. The y-axis shows the profit that results from the boundary being placed at the different points in the list (e.g. targeted marketing is directed at subjects above the line). Initially the profit rises more steeply (as a working classifier would rank positives, bringing profit, at the top) and starts falling once many negatives start being included above the boundary (these are false positives, incurring cost). In the picture, classification model 1 is good at identifying the instances with the highest probabilities, while classification model 2 has better performance in identifying the most highly probable 50% of instances.

Receiver Operating Characteristics (ROC) & Area Under Curve (AUC)

ROC has its origins in WW2, whence it was used as a measure of radar performance.



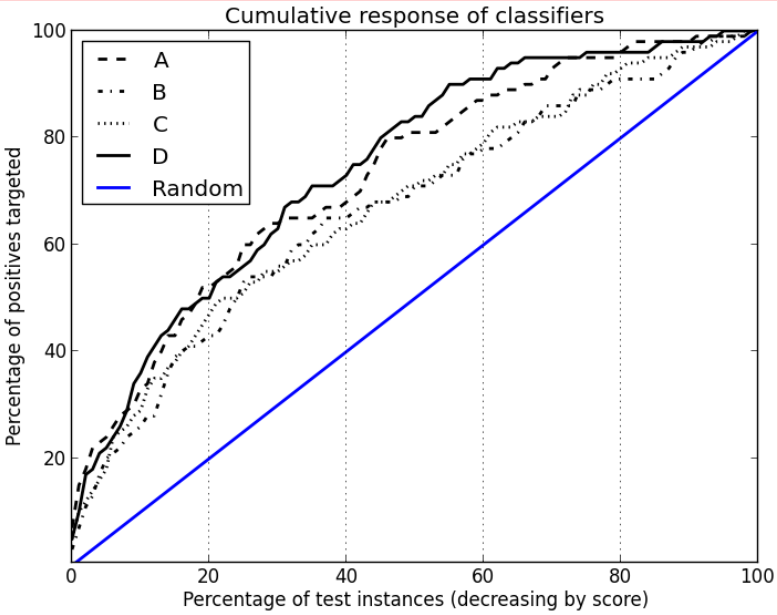
Each point in an ROC graph [left] corresponds to a contiguous portion, extending from the top, of a ranked instance list. The x-component of the point is the part (out of 1) of the data set's **true positives** contained in the list portion corresponding to the point. The y-component is the same part for **false positives**.

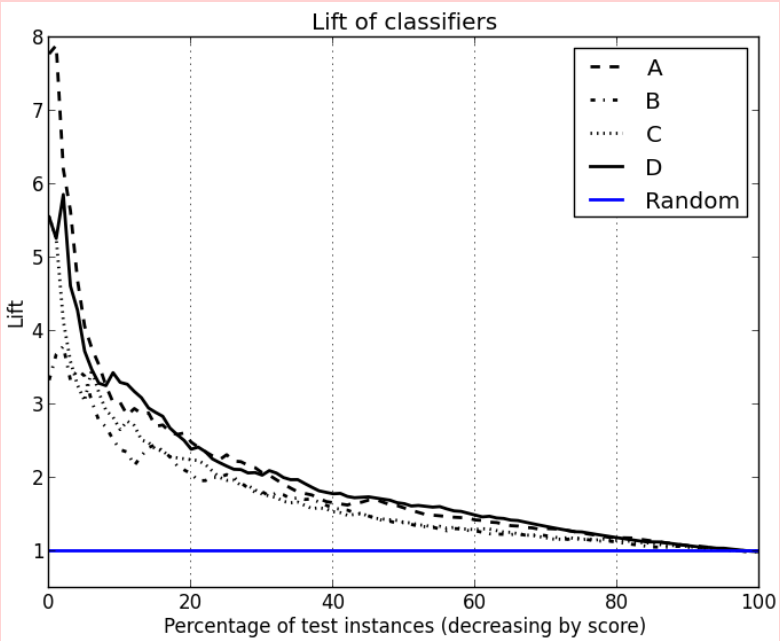


An ROC curve [right] is a systematic plot of points like those in the ROC graph together with a connecting line, for all possible contiguous portions, extending from the top, of the ranked instance list (starting with an empty list and concluding with the entire list).

ROC curves provide a measure of classifier performance that is **independent of class priors and of the cost-benefit matrix**. The closer the curve is to the two-straight-segment-line (0,0)-(0,1)-(1,1) the better the performance of the classifier, indicating more positives placed towards the top of the ranked list. The area under the ROC curve (AUC) provides a numeric measure of this quality: a value of 1 indicates a perfect classifier while a value of 0.5 correpsonds to random classification.

Cumulative response and lift curves



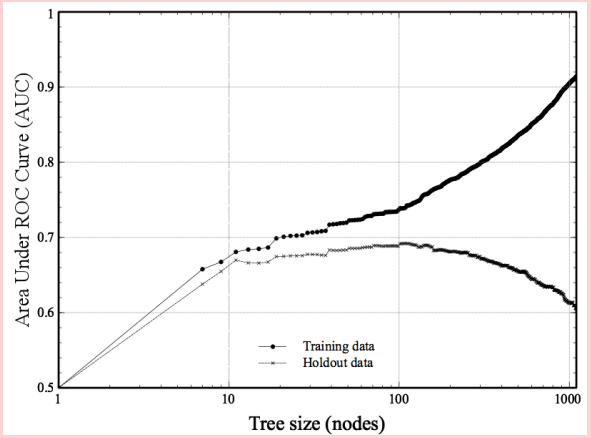


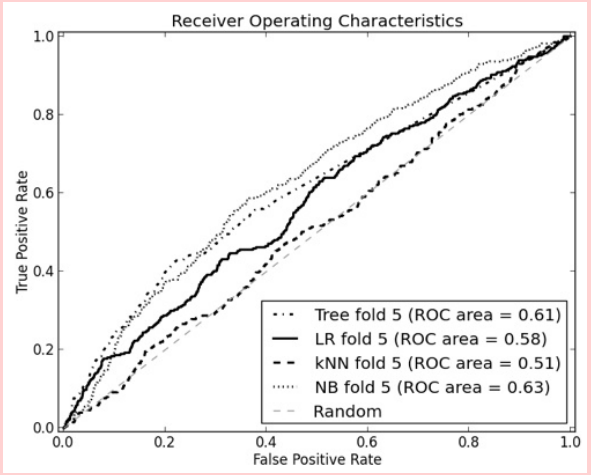
A **cumulative response curve** [left] is similar to an ROC curve but the x-axis, instead of representing the part of all false positives found in the cut-off portion of the ranked instance list, represents the percentage of the entire list of instances that is contained in the portion of the list above the cut-off point. This means that the shape of the cumulative response curve, while somewhat easier to understand for a non-technical audience, depends on the class priors and is thus not as pure a measure of a classifier's performance.

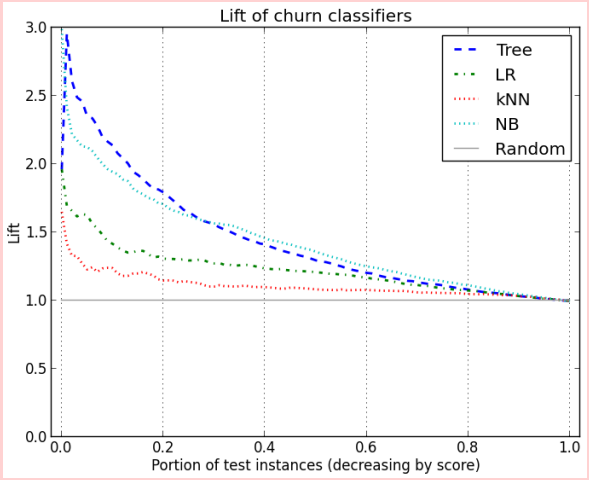
Every cumulative response curve (CRC) has a corresponding **lift curve** [right] which has the same values on the x-axis as the CRC and shows the ratios of the CRC y-values to the CRC y-values for random classification (this means that the lift curve for random classification is a constant 1).

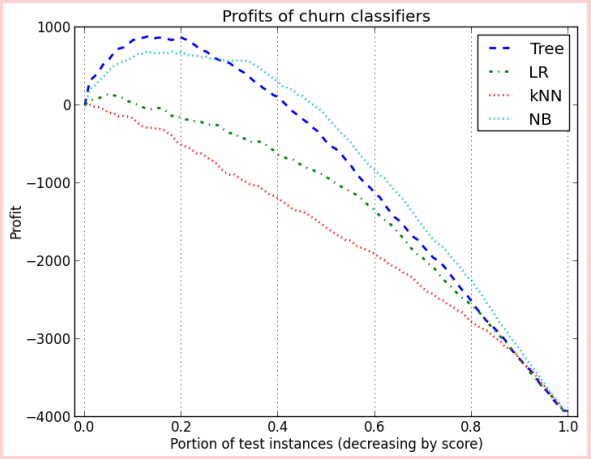
Example

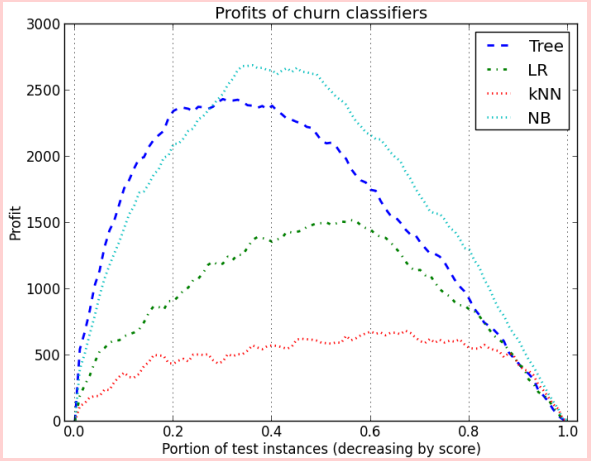
KDD Cup 2009 - French Telecom company Orange data (source: DSB)











Model	Accuracy (%)	AUC
Classification Tree	91.8 ± 0.0	0.614 ± 0.014
Logistic Regression	93.0 ± 0.1	0.574 ± 0.023
k-Nearest Neighbor	93.0 ± 0.0	0.537 ± 0.015
Naive Bayes	76.5 ± 0.6	0.632 ± 0.019

Although kNN produces considerably better accuracy than the Naïve Bayes classifier, the confusion matrices and diagrams show that kNN acts practically as a base-rate classifier and is in fact not the better of the two models.

Naïve Bayes Confusion Matrix

	p	n
Y	127 (3%)	848 (18%)
N	200 (4%)	3518 (75%)

k-NN Confusion Matrix

	p	n
Y	3 (0%)	15 (0%)
N	324 (7%)	4351 (93%)

Fitting graphs

- *Overfitting* is the application of model-building procedures to an extent that causes the created model to include properties specific to the training data, in addition to those actually pertinent to a general model. With a mathematical function overfitting may occur through the addition of too many attributes: more attributes means more dimensions and a better fit, but to the detriment of general applicability (more variance and more bias).
- Overfitting can be avoided by using *holdout* data to test that the accurracy of a model when used on new data matches, or is close to, the accuracy of the model when used on training data.

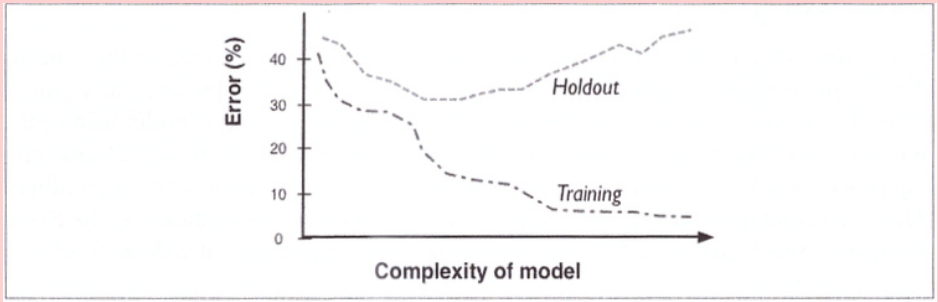


Figure 5-1. A typical fitting graph. Each point on a curve represents an accuracy estimation of a model with a specified complexity (as indicated on the horizontal axis). Accuracy estimates on training data and testing data vary differently based on how complex we allow a model to be. When the model is not allowed to be complex enough, it is not very accurate. As the models get too complex, they look very accurate on the training data, but in fact are overfitting—the training accuracy diverges from the holdout (generalization) accuracy.

Source: [DSB]