Topic 1. Overview of the Course

Case 1: Credit Scoring

using Public and Private Information

Reading Assignment

Berry and Linoff (2000)

- Pages 111–120. Decision trees.
- Pages 121–128. Neural networks.
- Pages 213–221. Boosting.

Topic Objective

Introduce the main ideas of machine learning.

- Tools
  - Regression – a model based method.
  - Nearest neighbors – a smoothing method.
  - Decision trees – automated data partitioning.
  - Neural nets – automatic feature detection and thresholding.
- Performance criteria – the loss function.
- Performance assessment – hold out samples and cross validation.
- Boosting – combining tools.

The context is classification but, as we shall see, the ideas are the similar in prediction. Some are exactly the same because many tools use prediction to classify. All data mining approaches, however different they may seem, are actually some blend of these ideas.

Case 1: Credit Scoring

The first input measure, x_1, is a FICO credit score purchased from Fair, Isaac & Company scaled so that $0 < x_1 < 1$. This is public information available to all lenders.

The second input measure, x_2, is a profitability index computed from a transactions database containing account balances, previous loan history, etc., scaled so that $0 < x_2 < 1$. This is private information.

The output variable, y, is coded as $y = 1$ in case of default and $y = 0$ if the loan is paid in full.

Types of Variables

Not every tool works gracefully with every type of data.

See Berry and Linoff pp. 153 – 157.

Here are the flavors that data comes in:

Types of Variables (continued)

Quantitative variables are variables for which order and distance are meaningful.

Test scores are an example. Order obviously matters. Distance matters because if $T_2 - T_1 > T_3 - T_1 > 0$, then T_2 exhibits better mastery of the subject than T_3.

Interval variables are quantitative variables such as dates for which differencing is meaningful but other arithmetic operations are not.

True numerics are quantitative variables such as dollars that support a full range of numeric operations.

Types of Variables (continued)

Qualitative variables are variables for which distance is not meaningful. Order may or may not matter.

Synonyms for qualitative variable are categorical variable, discrete variable, class variable, symbolic variable, and factor.

An example of an ordered categorical variable is T-shirt sizing: S, M, L, XL, XXL. An example of an unordered categorical variable is quality measured as "defective" and "nondefective".

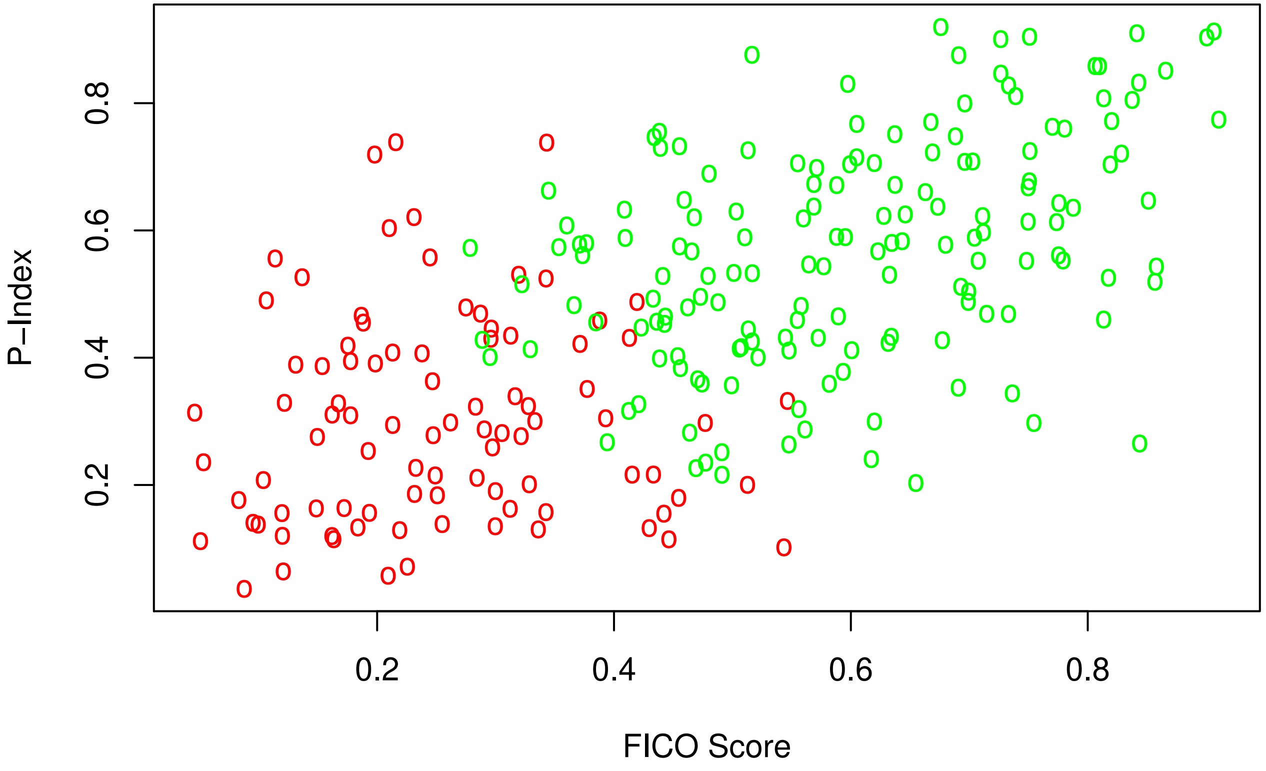
Qualitative variables are often coded as 0,1 for binary categories and 1,2,3,... for richer categorizations to have a convenient machine representation. For ordered qualitative variables this can be done so the numbers are meaningful as ranks.

Case 1 (continued)

The two input measures x_1 (FICO Score) and x_2 (P-Index) are quantitative; the output measure y (Default) is qualitative.

The training set is 250 cases consisting of the loan application forms from loans that have either defaulted or that have been paid. The FCIO Score and P-Index are noted on the loan application forms.

Fig 1. The Training Data



Defaulted loans shown in red; loans paid in full shown in green.

Linear Regression Classification

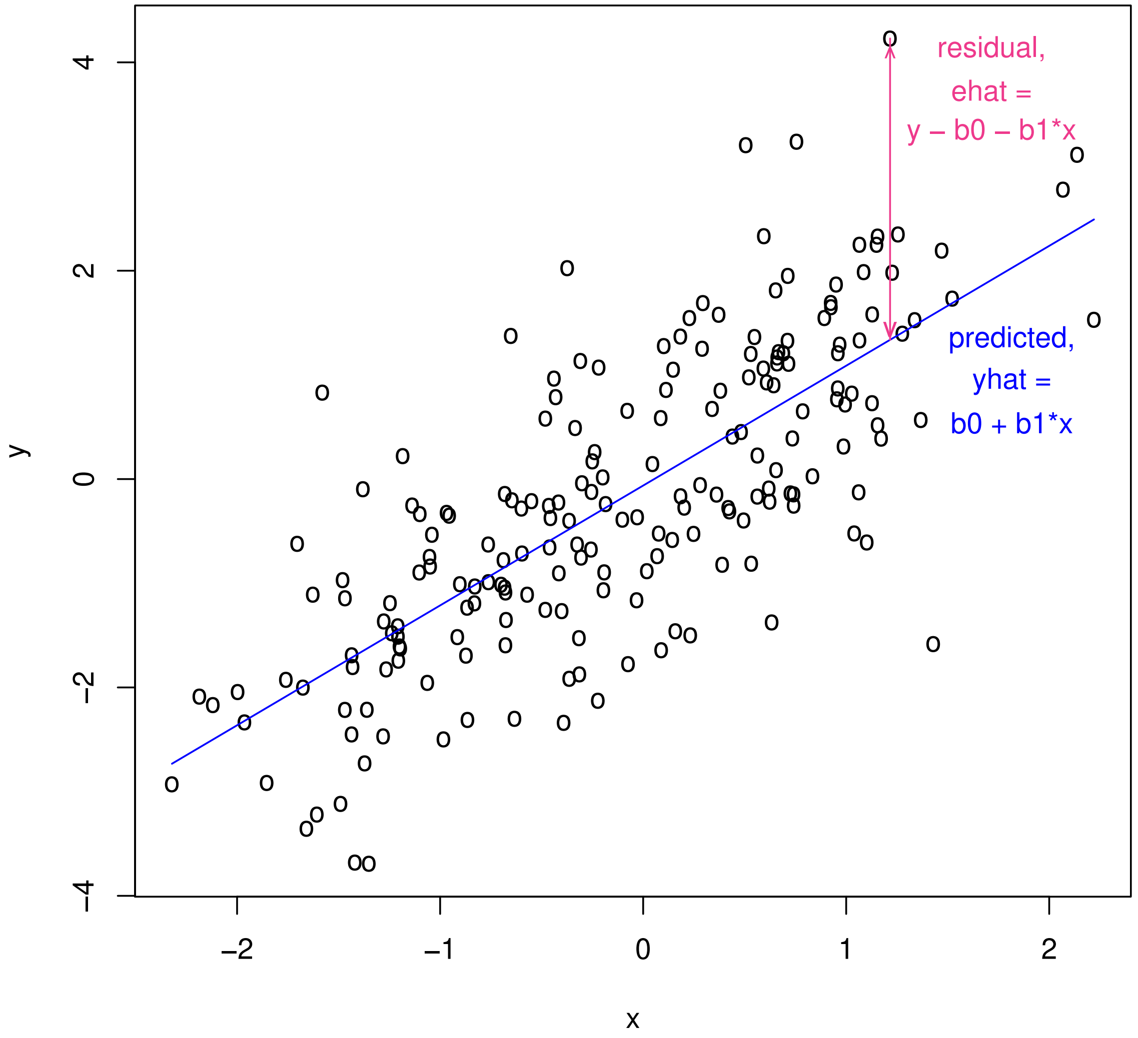
Least squares or linear regression classification treats the 0's and 1's as if they were a quantitative dependent variable – as y's – and finds the best fitting linear function of the features – the x's – by minimizing a squared error loss function within the training data set.

To classify a case, a prediction is made from its features, and the case is classified as 0 or 1 depending on whether the prediction is greater or less than $\frac{1}{2}$.

This amounts to dividing cases into two groups by drawing a line through feature space.

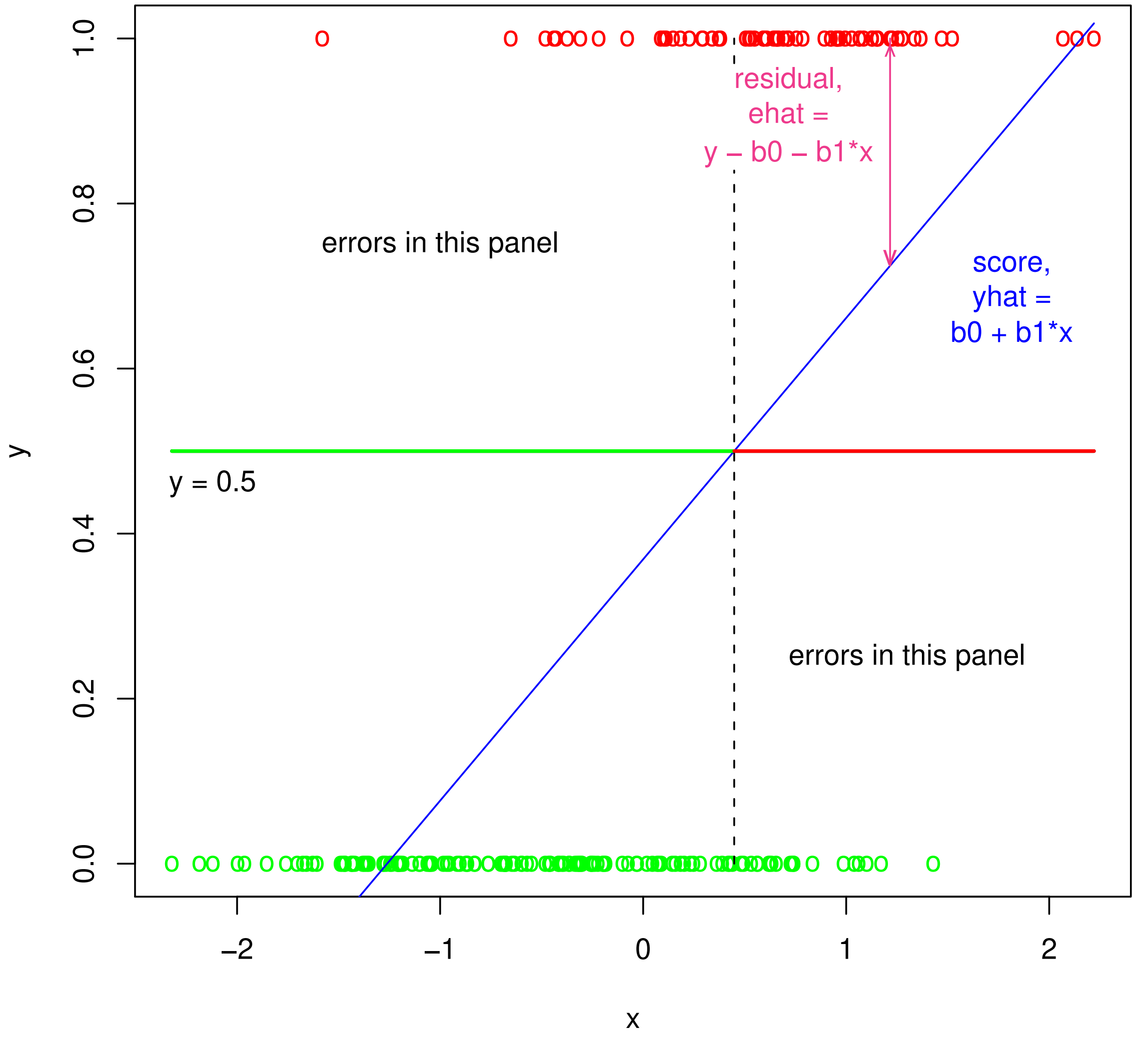
Details follow:

Fig 2. Linear Regression



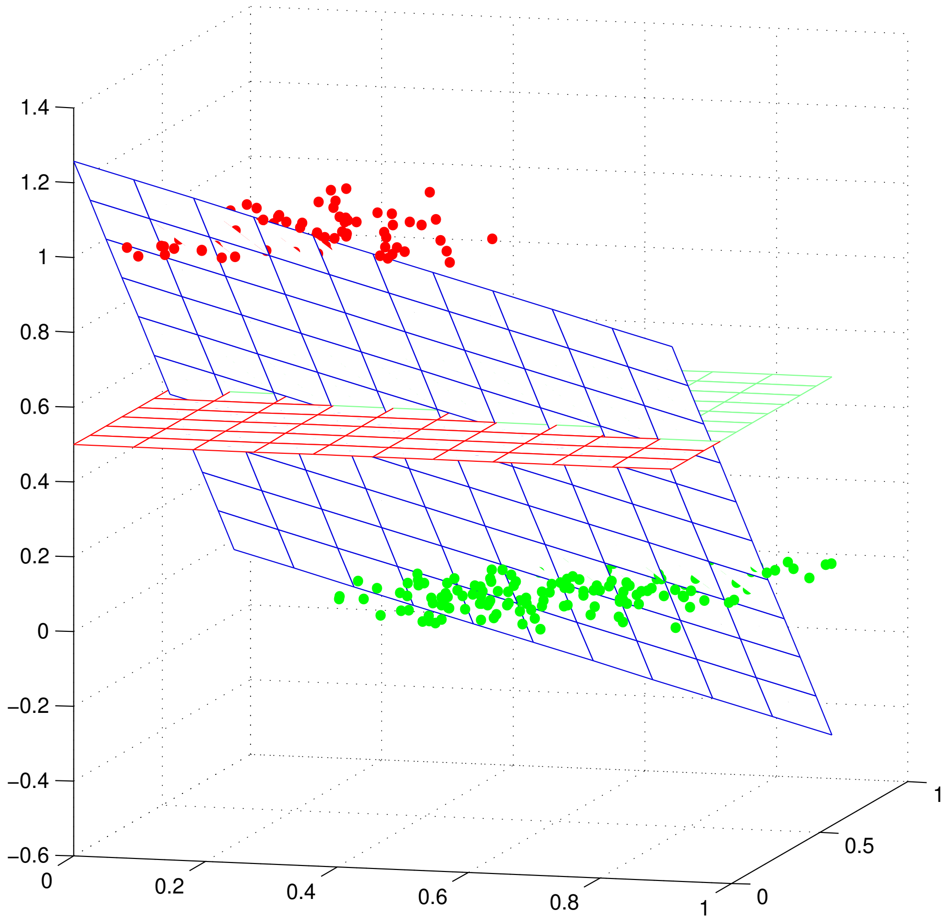
This should be familiar from your statistics course.

Fig 3. Linear Regression Classification



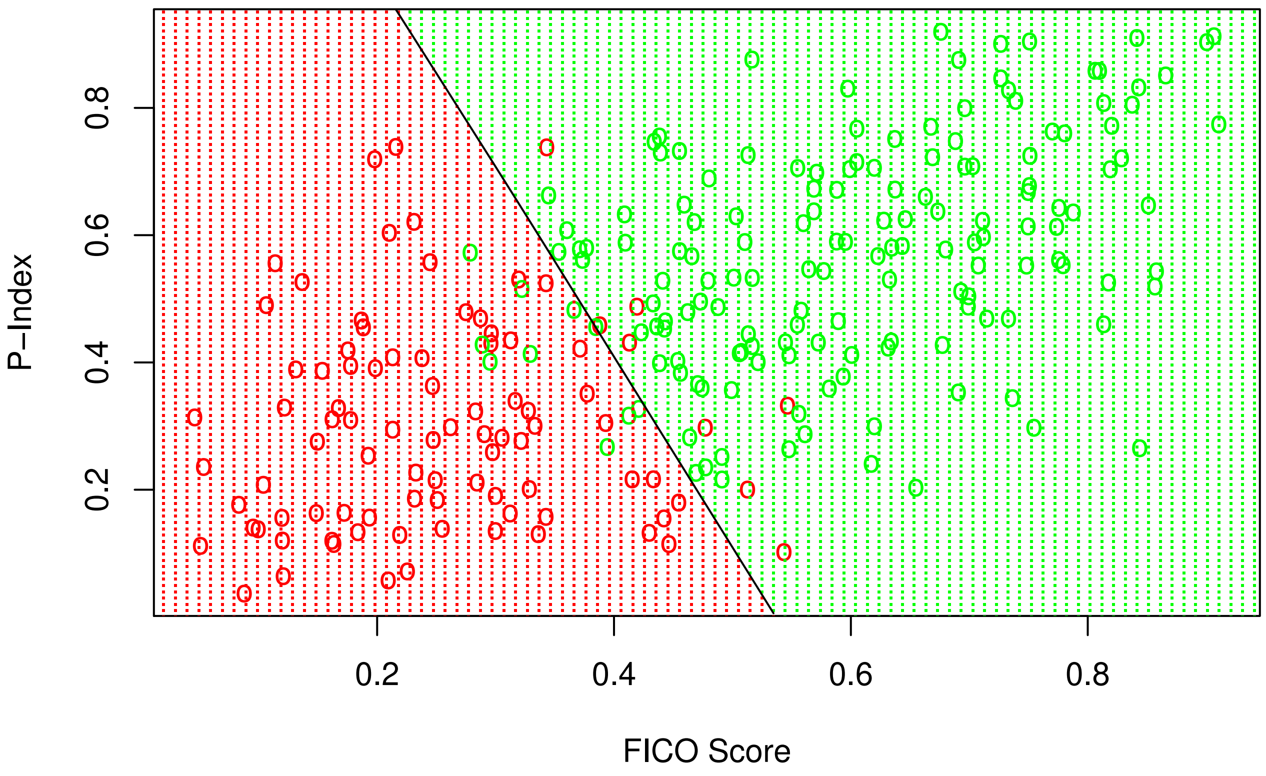
In a classification regression the dependent variables are 0's and 1's.

Fig 4. Loan Case Linear Regression Classification



The regression surface for the loans case, defaults are red.

Fig 5. Loan Defaults Linear Regression Classification



This is Figure 4 projected to two dimensions. Shading indicates the classification regions; red predicts default and green predicts payment. The training set is shown as circles.

Linear Regression Classification: The Math

- Minimize: $\sum_{i=1}^{n} (y_i - b_0 - b_1 x_{i1} - b_2 x_{i2})^2$
- Solution: $(\widehat{b}_0, \widehat{b}_1, \widehat{b}_2) =$ (1.257, -1.414, -0.472)
- Case i's Score: $\widehat{y}_i = \widehat{b}_0 + \widehat{b}_1 x_{i1} + \widehat{b}_2 x_{i2}$
- Classification Rule: Predict that case i with features x_{i1}, x_{i2} will default if $\widehat{y}_i > 0.5$.

Scoring

A case i with FICO Score $= x_{i1}$ and P-Index $= x_{i2}$ in Figure 5 is classified as a 0 or 1 by computing the predicted value

$$\widehat{y}_i = 1.257 - 1.414x_{i1} - 0.472x_{i2}$$

and then thresholding: 1 if $\widehat{y} > 0.5$, 0 else.

The predicted value $\widehat{y}_i$ that is assigned to case i is called a score. The process of assigning scores to cases is called scoring. (Berry and Linoff p. 60)

All tools score the data in some way.

In most applications the scores, which rank the cases, are more useful than the thresholded values.

Logistic Regression

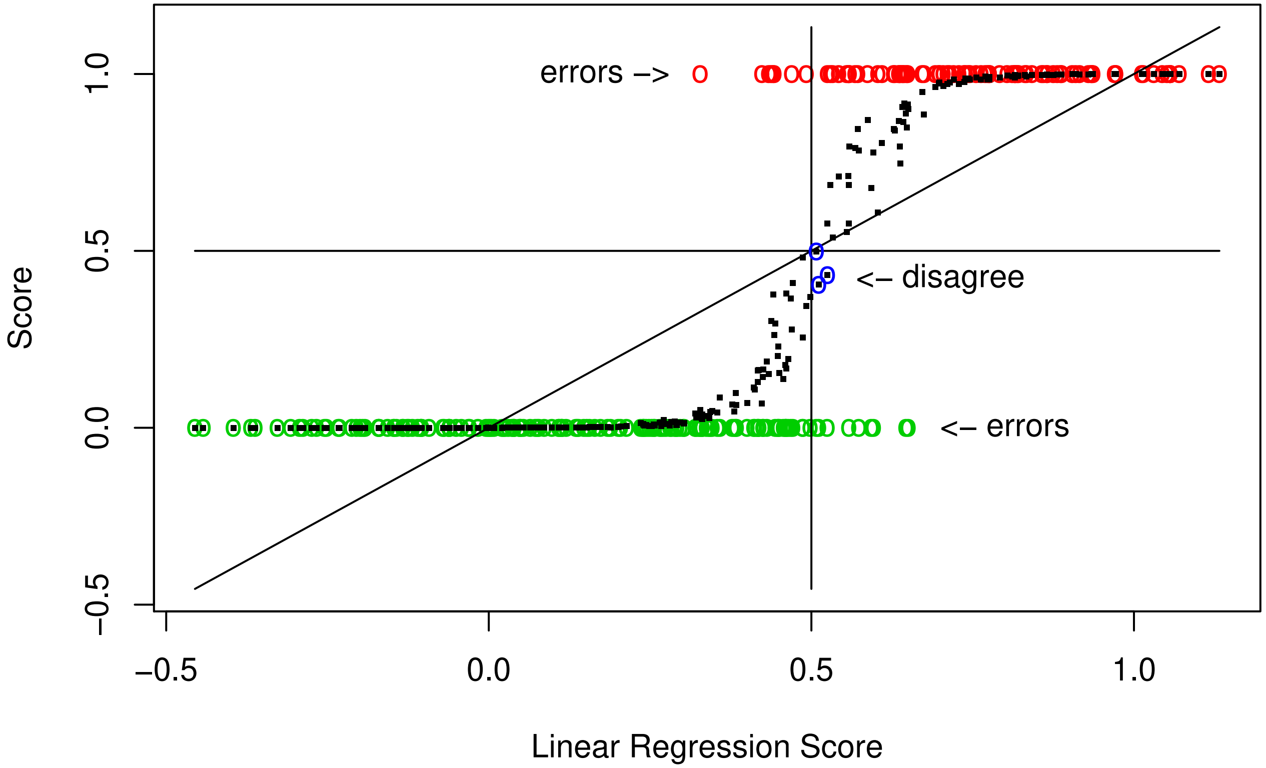
Logistic regression, which we will study in more detail in Topic 8, is the same as linear regression except that the scores that logistic regression produces are directly interpretable as probabilities. For example, if logistic regression scores case i as 0.75 it means that the probability of default is 75%.

This interpretability can be useful in explaining results to clients.

The ranking of the cases produced by logistic regression and linear regression will not differ much. Linear regression is more stable numerically and cheaper to compute and therefore a bit safer if interpretability is not essential.

The next figure compares linear regression and logistic regression.

Fig 6. Linear and Logistic Regression Scores



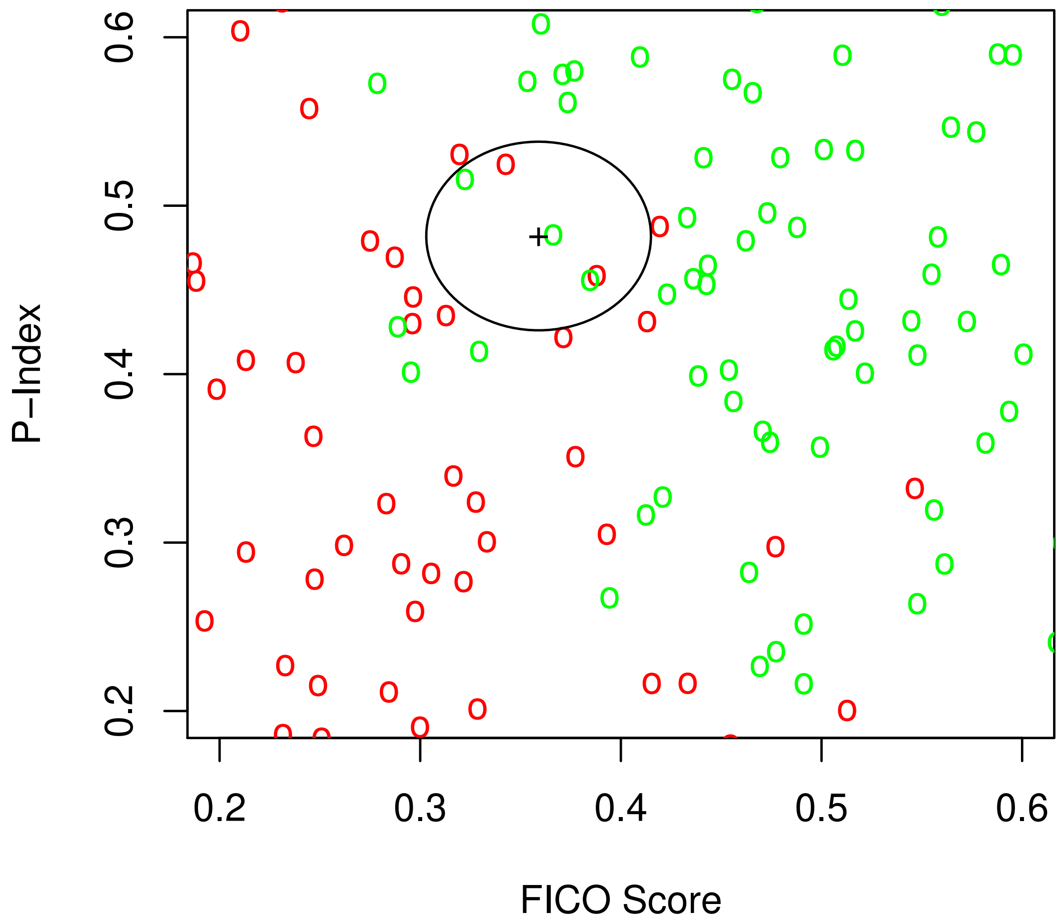
The 45 degree line is the linear regression score plotted against itself. The black dots are the scores assigned by logistic regression. The horizontal line is the threshold value of 0.5 that applies to both sets of scores. The red and green circles are the data, with red indicating default. Circles in the upper left and lower right quadrants are linear regression classification errors. Dots circled in blue are cases where logistic regression classifies differently than linear regression. Linear regression makes 17 classification errors; logistic regression makes 16 classification errors. The correlation of the ranked scores is 0.998.

Nearest Neighbor Classification

1. To classify a case with features $x = (x_1, x_2)$ make x the center of a circle and increase the radius of the circle until it encloses the features of exactly K cases of the training data set.

2. Classification is like voting: If a majority of the cases within the circle are defaults, then the case at the center of the circle is classified as a default.

Fig 7. Nearest Neighbor Classification



The black cross is the point to be classified. The circle is made just large enough to enclose $K = 5$ points. Majority rules: 3 green vs. 2 red means green wins.

Nearest Neighbor Classification

Mathematical Representation:

$$y(x) = \frac{1}{K} \sum_{x_i \in \mathcal{N}_K(x)} y_i$$

where $\mathcal{N}_K(x)$ is the neighborhood of x containing the K closest points x_i in the training sample.

Predict that a case with features x will default if $y(x) > 0.5$.

Role of K in Nearest Neighbors

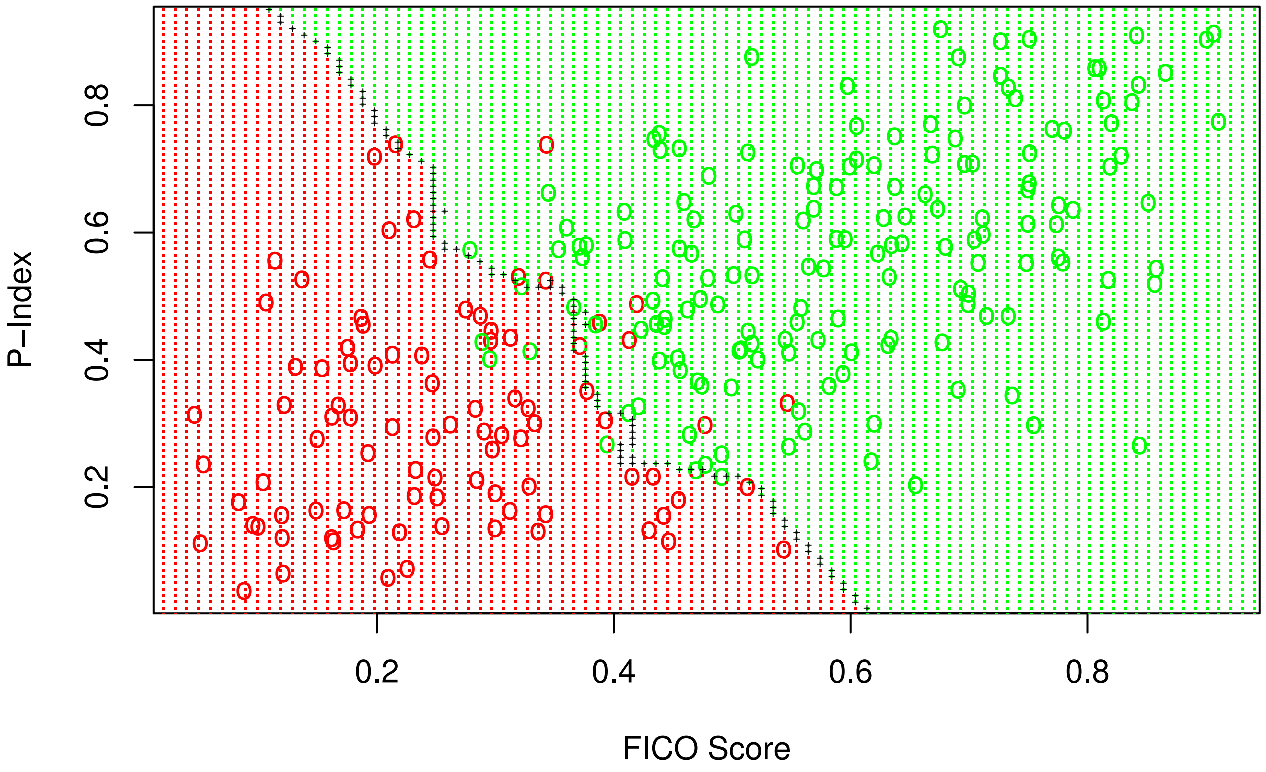
The formula is an average of K items so it would seem that:

1. If K is large, feature space will be divided into two groups with a smooth boundary.

2. If K is small, the boundary will be a ragged mess and islands of classification are possible.

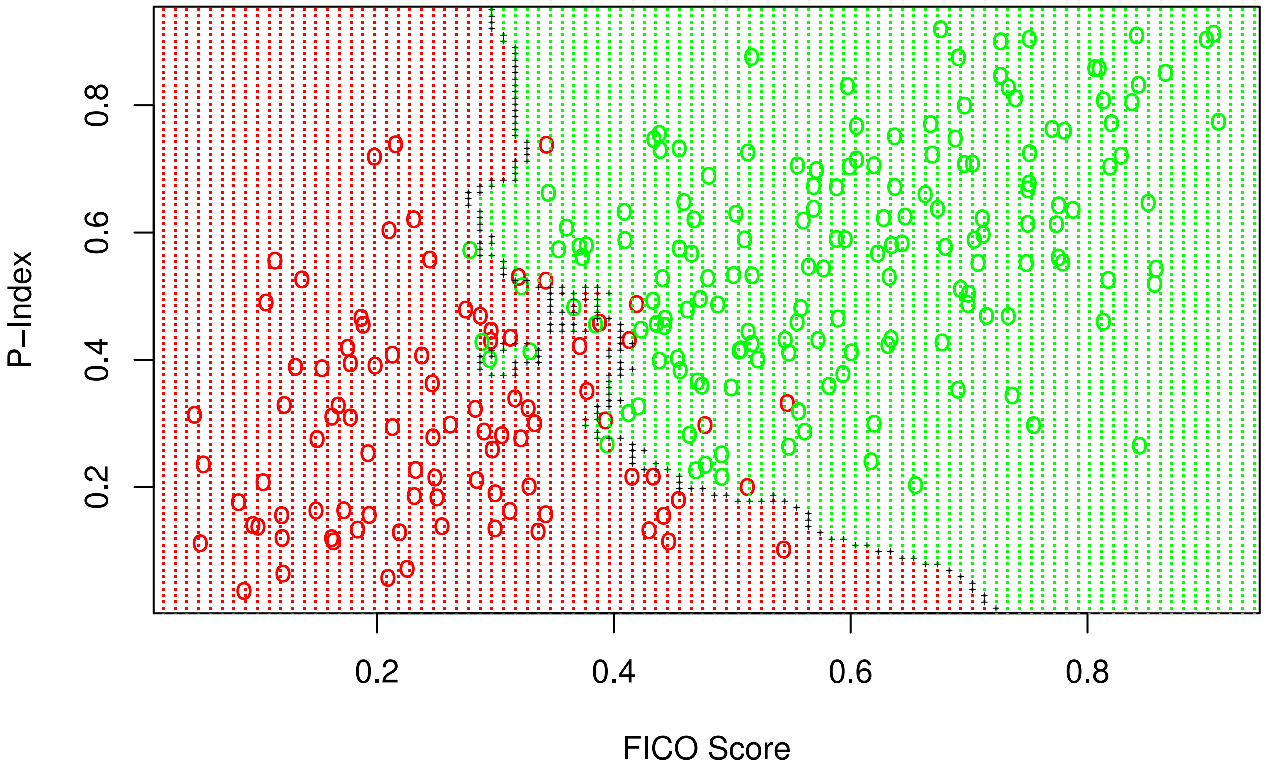
Let's try $K = 15$ and $K = 5$ to see what happens.

Fig 8. Nearest Neighbor Classification, $K = 15$



Shading indicates the classification regions; red predicts default and green predicts payment. The training set is shown as circles.

Fig 9. Nearest Neighbor Classification, $K = 5$



Shading indicates the classification regions; red predicts default and green predicts payment. The training set is shown as circles.

How Many Neighbors Should There Be?

- How to choose a tuning parameter like K in theory is the bias-variance trade-off problem (statistical terminology) or the learning-generalization trade-off problem (computer science terminology) – Topic 2.

- The best way to choose K in practice is to get more data and check performance in this new data that was not used to train the tool.

- But we want answers now using the data we have now. Waiting for new data to arrive is not a reasonable suggestion.

- There is a solution ...

There Are Actually Two Questions

1. Where is this new data to be found right now?

2. How is performance to be assessed in this new data?

Let us consider the second question first.

Loss Functions

The loss function $L(y, \widehat{y})$ tells us how much errors cost us.

Here, y is what is observed and $\widehat{y}$ is what is predicted.

There are only four outcomes, two are successes and have zero loss and the other two are failures and have positive loss:

$$0 = L(0, 0) \text{ when } y = 0 \text{ and } \widehat{y} = 0$$

$$0 = L(1, 1) \text{ when } y = 1 \text{ and } \widehat{y} = 1$$

$$l_0 = L(0, 1) \text{ when } y = 0 \text{ and } \widehat{y} = 1$$

$$l_1 = L(1, 0) \text{ when } y = 1 \text{ and } \widehat{y} = 0$$

The prediction $\widehat{y}$ is computed from features x; we write $\widehat{y}(x)$ when this fact needs emphasis.

How To Assess Performance

Call this new data the validation data.

Add up all the losses in the validation data set:

$$\text{Total Loss} = \sum_{i=1}^{n} L[y_i, \widehat{y}(x_i)]$$

$$\text{Average Loss} = \frac{1}{n} \sum_{i=1}^{n} L[y_i, \widehat{y}(x_i)]$$

If $l_0 = l_1 = 1$, the average loss is the proportion of cases that are missclassified aka the classification error rate.

How Choose K

Choose the K that minimizes Loss in the new data.

Where Does the New Data Come From?

If the data set is large (100,000 or more), which is the usual case in data mining, divide the data randomly into two data sets: the training sample and the new data, called the validation sample.

Some divide into three samples: training, validation, and test.

The test sample is used as a final check to compare tools because the validation sample is actually used to train some of the tools; i.e. it is used to choose K for nearest neighbors.

I like to use 60%, 40% for training and validation.

Validation Sample

Later I want to compare the performance of the tools considered to the theoretical optimum.

To do this, the data has to be simulated because I need to know what actually generated the data to know what is the optimal classification scheme.

Because the data are simulated, I can have as big a validation sample as I want.

I generated a validation sample of 100,000 to get reasonably accurate estimates of loss.

Table 1. Classification Error Rates

Method	Error Rate
Nearest Neighbor, $K = 15$	0.07697
Linear Regression	0.07854
Nearest Neighbor, $K = 5$	0.07943

$K = 15$ is preferred.

Accuracy in Validation Samples

Accuracy in validation samples scales as the square root of sample size because Average Loss is a sample mean:

$$\text{Average Loss} = \bar{\mathcal{L}} = \frac{1}{n} \sum_{i=1}^{n} L[y_i, \widehat{y}(x_i)]$$

$$\text{Variance} = s^2 = \frac{1}{n-1} \sum_{i=1}^{n} \left(L[y_i, \widehat{y}(x_i)] - \bar{\mathcal{L}} \right)^2$$

$$\text{Accuracy} = \pm 1.96\, s \frac{1}{\sqrt{n}}$$

Accuracy in This Validation Sample

A Bernoulli random variable takes the value 1 with probability p and 0 with probability $q = 1 - p$. A Bernoulli random variable has variance pq.

The Table 1 loss function is the average of Bernoulli random variables and each line has $\bar{\mathcal{L}} \doteq 0.08$. Putting $p = 0.08$ and using pq as the variance, which gives better accuracy for Bernoulli random variables than s^2, we apply the formulas from the previous slide to get

$$1.96\, s \frac{1}{\sqrt{n}} = (1.96)(0.2713)(1/316.2) = 0.001682.$$

In Table 1, we are accurate to within ± 0.002.

Cross Validation

If there is too little data to waste on a validation sample, proceed as follows.

- Make sure that the data are in random order.
- Hold out the first 10% of the cases, train on the remaining 90%, and compute the loss in the hold out sample.
- Hold out the second 10% of the cases, train on the remaining 90%, and compute the loss in the hold out sample.
- $\vdots$
- Hold out the last 10% of the cases, train on the remaining 90%, and compute the loss in the hold out sample.
- Average the ten losses just computed.

Cross Validation

Cross validation as described on the previous slide is called "ten part cross validation."

One can do "five part" or "twenty part" or whatever; most seem to prefer "ten part."

The extreme form is to hold out one case, train on the remainder, compute the loss for the one held out case, do this for every case in the data, and then average all the losses.

This is called "cross validation" with no adjective.

Many tools have cross validation built into them. In the cases I present in class, I shall nearly always suppress cross validation and use a validation sample.

Asymmetric Loss

Nearest neighbors trains for the case that $l_0 = l_1$. If, e.g., a loan default is twice as costly as losing a customer denied credit, then $l_0 = 1$ and $l_1 = 2$ are more reasonable. To adjust how nearest neighbors trains:

1. If the implementation allows the use of weights
   - weight defaults by $1.4 = \sqrt{2}$ if the implementation weights by the square
   - weight defaults by 2 if the implementation uses weights as given
2. Otherwise
   - replicate each row of the training data that defaults if the training data set is small
   - randomly discard $\frac{1}{2}$ of the cases in the the training data that do not default if training data set is large

Decision Trees

Decision trees are probably already familiar to you. The idea is straightforward:

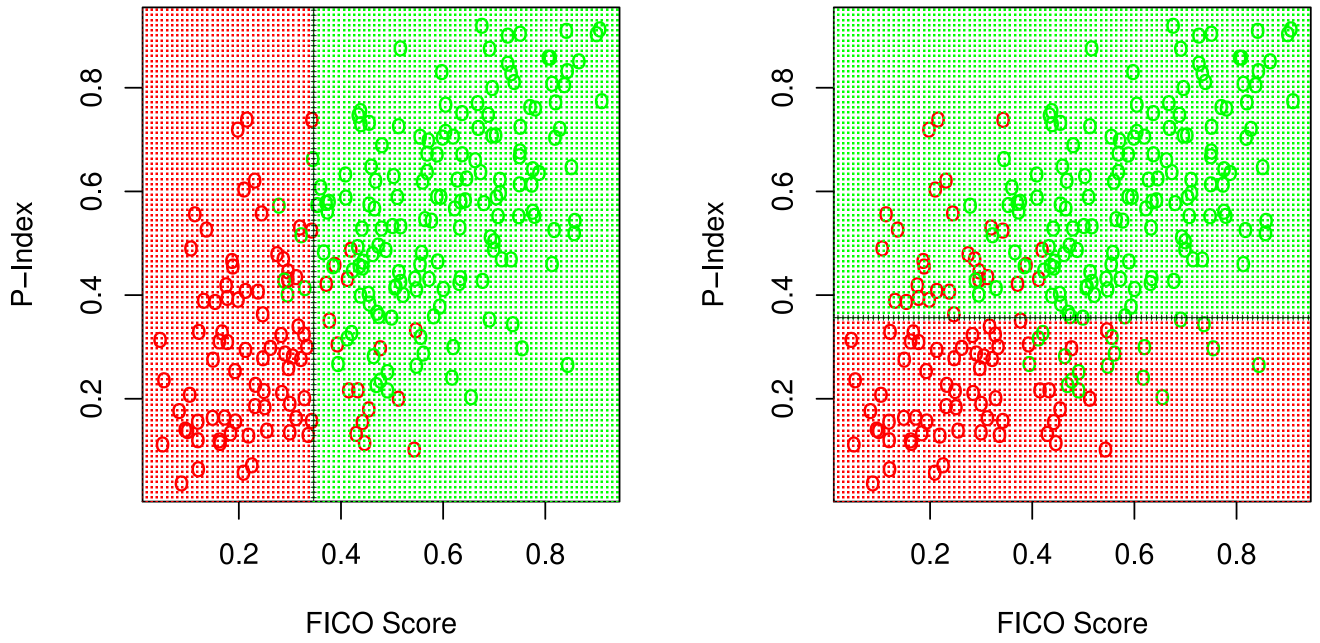
1. Start with feature x_1 and divide the sample into two groups depending on whether the feature is to the left, $x_1 < c$, or the right, $c < x_1$, of the cut c. In each group, majority rules, i.e. if there are more defaults to the right, every point that satisfies $c < x_1$ is classified as default.

2. Compute the loss over the sample; try all possible cuts; remember the cut with smallest loss. Do this for every feature and choose the feature and cut with smallest loss.

3. Work down the tree, subdividing groups using the same procedure.

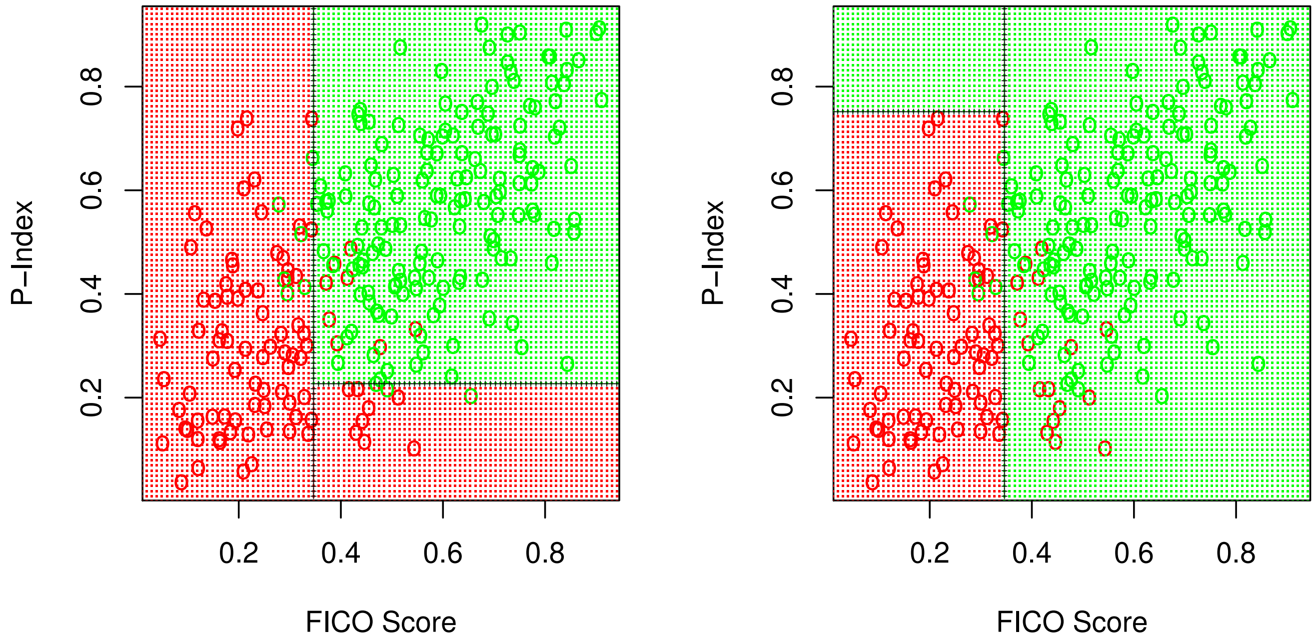
4. Stop when some measure of complexity such as the depth of the tree or the number of leaves is reached.

Fig 10. Decision Tree: First Cut



For the best cut on FICO Score there are 21 classification errors. For the best cut on P-index there are 47 classification errors. Therefore the first cut is on FICO score. By inspection one can see that the second cut will have to be on P-index.

Fig 11. Decision Tree: Second Cut



For the best cut on P-index of the group on the right there are 15 classification errors. For the best cut on P-index of the group on the left there are 21 classification errors. The tree that makes 15 classification errors is preferred.

Decision Trees

Decision trees are very popular; perhaps the most popular data mining tool.

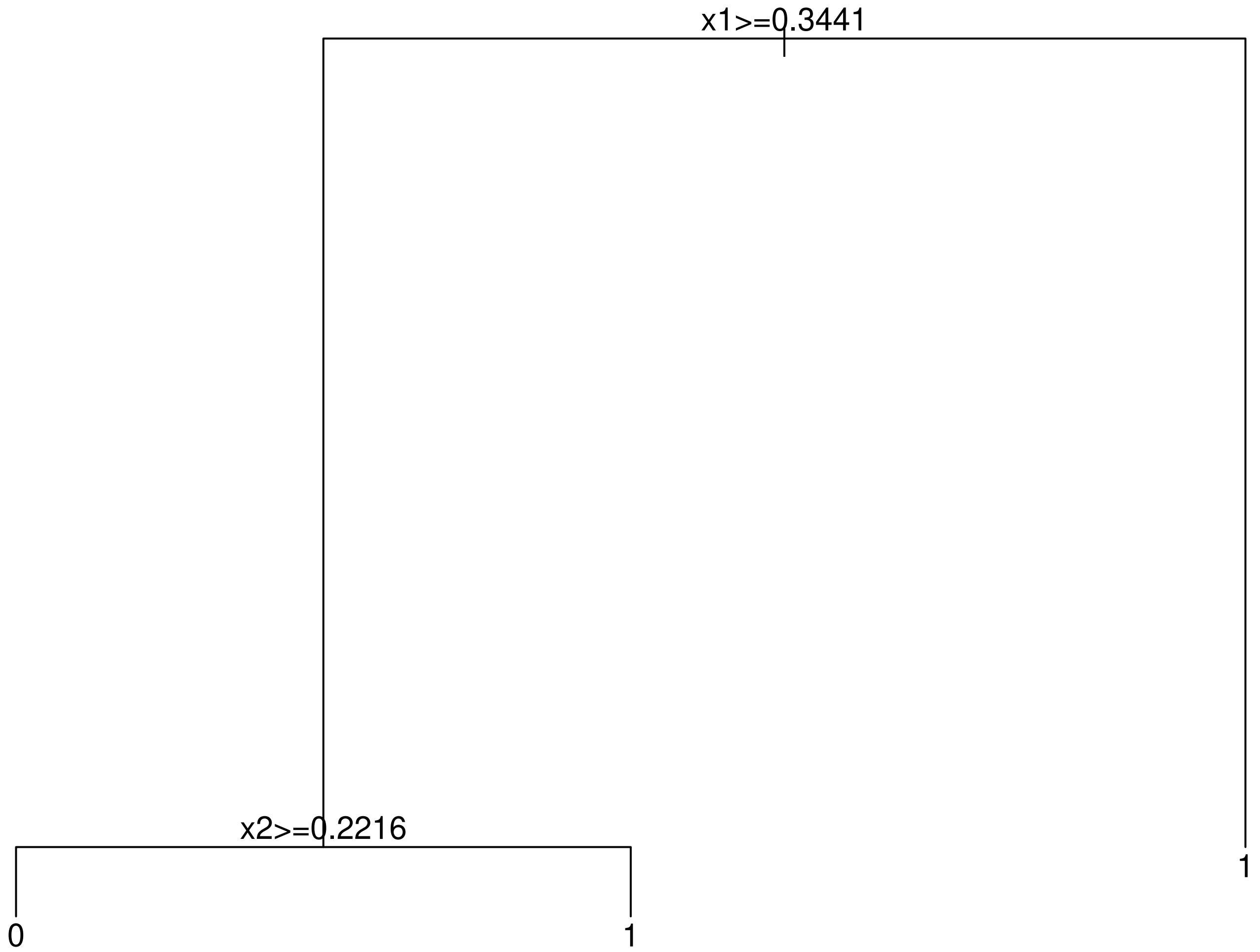
They are easy to explain and easy to use.

One learns something about the data from them: The important features are clearly identified.

For whatever measure of complexity, the best tree is chosen by either evaluating loss in a validation sample or by cross validation, exactly the same as for nearest neighbors. For example, if the measure of complexity is the depth of the tree, one checks loss for all possible depths in the validation sample and chooses the depth with smallest loss.

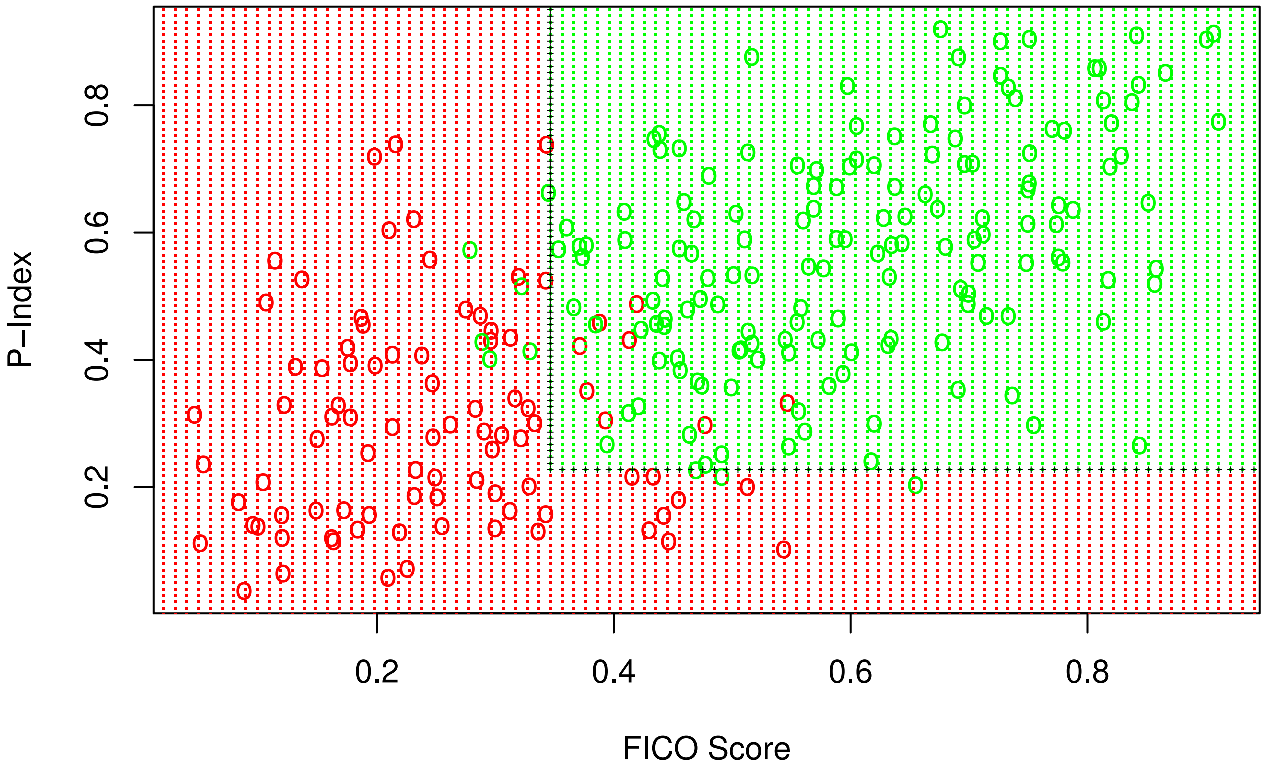
Our complexity parameter will be the minimum size of a split and we will try two of them, minsplit=20 (the default) and minsplit=15.

Fig 12. Decision Tree, minsplit=20



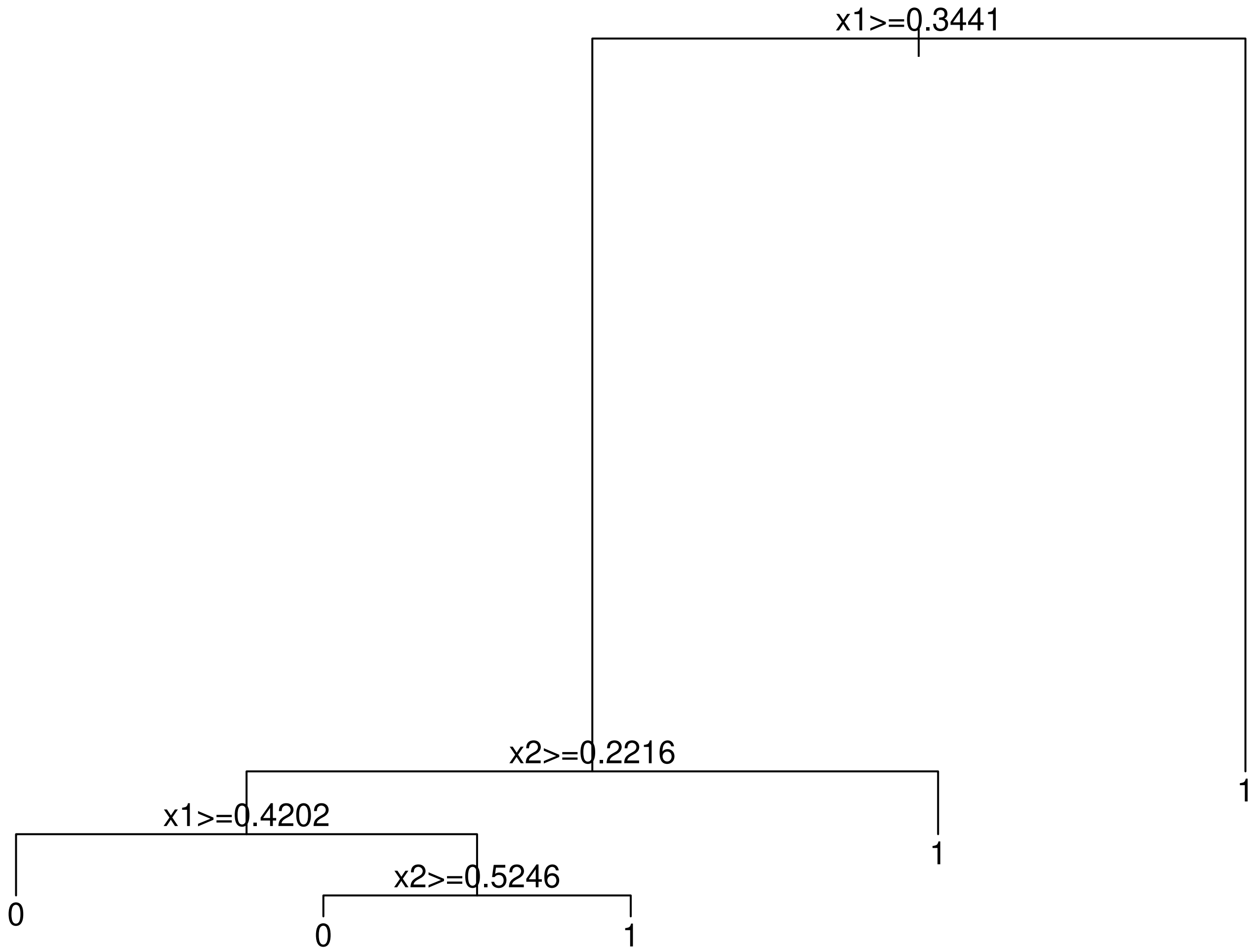
CART decision tree. The inequality refers to left branch of the tree; a leaf marked 1 indicates default; 0 indicates payment in full.

Fig 13. Decision Tree, minsplit=20



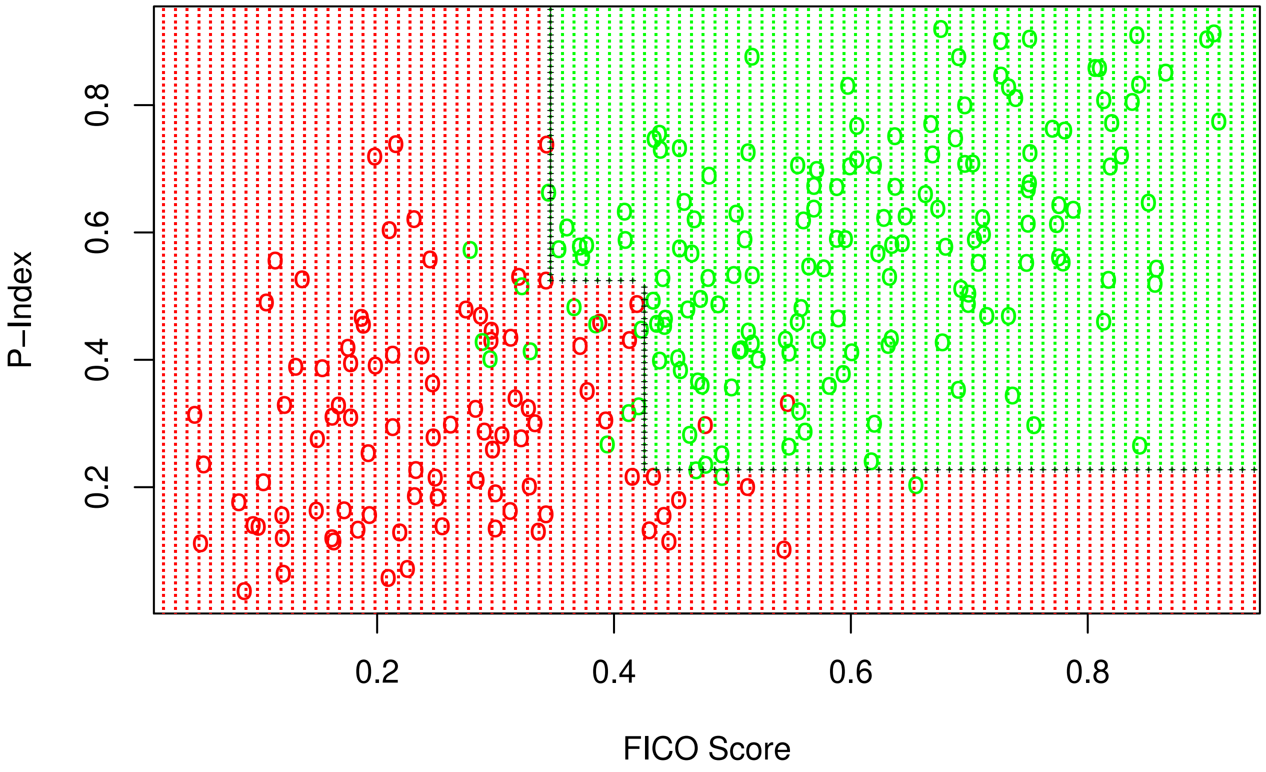
Shading indicates the classification regions; red predicts default and green predicts payment. The training set is shown as circles.

Fig 14. Decision Tree, minsplit=15



CART decision tree. The inequality refers to left branch of the tree; a leaf marked 1 indicates default; 0 indicates payment in full.

Fig 15. Decision Tree, minsplit=15



Shading indicates the classification regions; red predicts default and green predicts payment. The training set is shown as circles.

Table 2. Classification Error Rates

Method	Error Rate
Nearest Neighbor, $K = 15$	0.07697
Linear Regression	0.07854
Nearest Neighbor, $K = 5$	0.07943
Decision Tree, minsplit=20	0.08875
Decision Tree, minsplit=15	0.09149

Values within 0.002 can be regarded as the same.

Neural Nets

The physical structure of the brain has been known for centuries: It is a mass of neurons connected by dendrites that transmit electro-chemical signals.

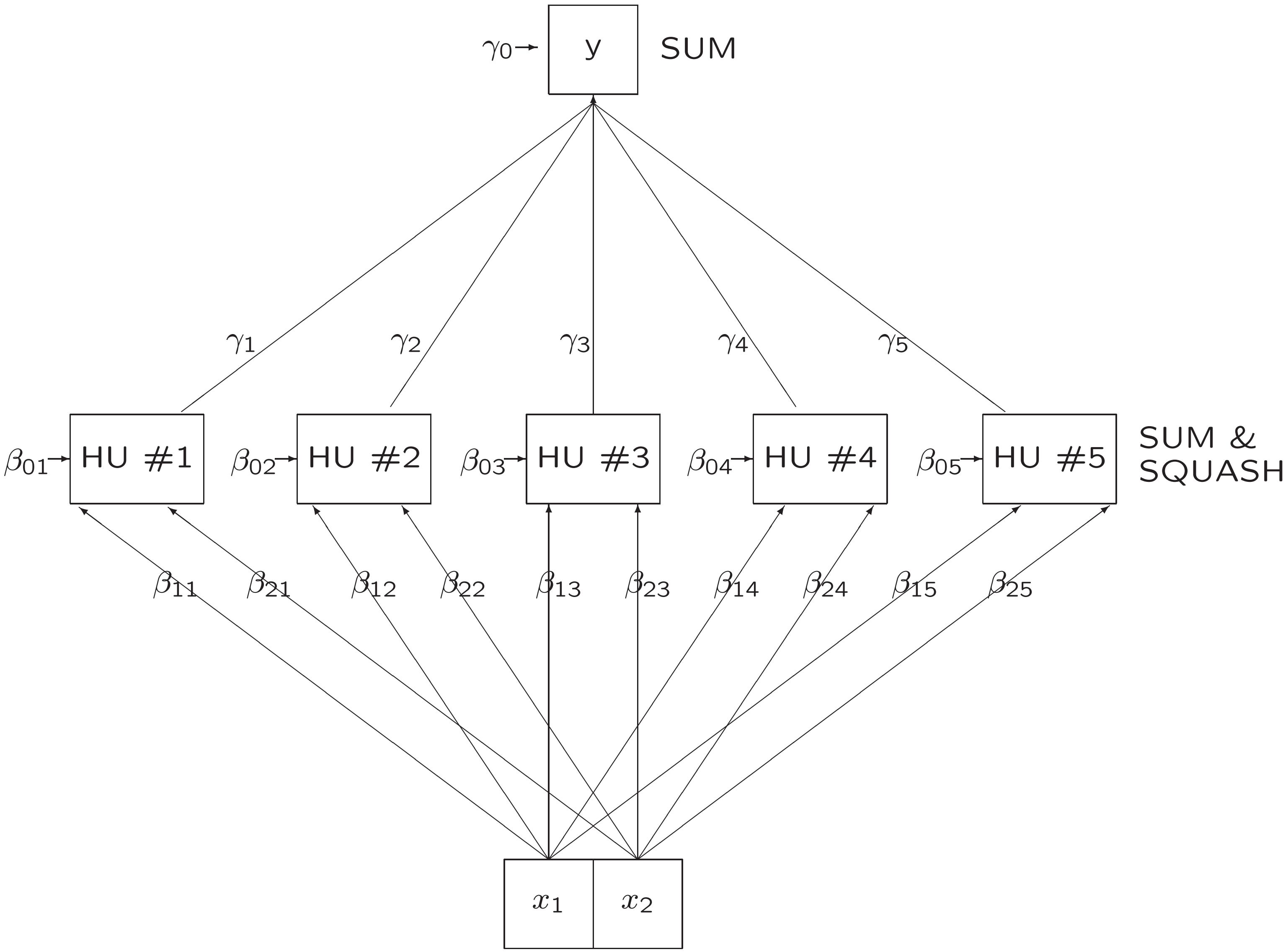
Neural nets were invented to show that this type of structure could exhibit intelligence.

The moment of triumph occurred when it was shown mathematically that such a structure could represent xor (exclusive or).

Since then they have been used for robotics, data analysis, text recognition, detection of chaos, etc.

One needs to look at both their diagrammatic and mathematical representation to understand them – next few slides.

Fig 16. Single Hidden Layer Neural Net, Five Hidden Units



What the Diagram Represents

Boxes are neurons and lines are dendrites.

The two boxes at the lowest level represent sensory neurons. They send signals of varying strength to the neurons above them. Signal strength is represented by the β_{ij}.

Each of these second level neurons additively combine the weighted signals, adding to them a bias represented by the β_{0j} . If the sum exceeds a threshold, it is passed on to the next higher level with varying strengths represented by the γ_j.

The top neuron additively combines these weighted signals, adding a bias γ_0. This sum may or may not be thresholded.

Shown is a single hidden layer feed forward neural net. One can have more hidden layers, feedback, etc. But for data analysis it can be proved that a single hidden layer feed forward net is adequate.

Single Hidden Layer Neural Net, Five Hidden Units

Mathematical Representation:

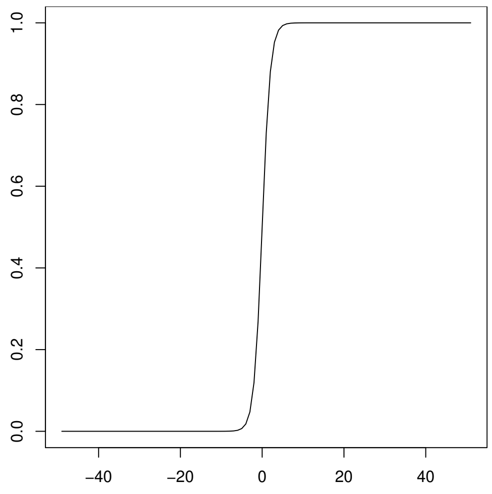
$$y = \gamma_0 + \sum_{j=1}^{5} \gamma_j \, S\left(\beta_{0j} + \beta_{1j} x_{1j} + \beta_{2j} x_{2j}\right)$$

Weights:

$$\gamma_0, \, \gamma_1, \, \beta_{01}, \, \beta_{11}, \, \beta_{21}, \, \ldots, \gamma_5, \, \beta_{05}, \, \beta_{15}, \, \beta_{25}$$

Squasher:

$$S(x) = \frac{\exp(x)}{1 + \exp(x)}$$



What the Mathematics Represents

The mathematical representation shows the summation and thresholding.

The threshold function is called a squasher in the neural net literature and is usually chosen to be a differentiable function as shown in the slide.

From a statistical perspective, a neural net can be viewed as a nonlinear regression that can fit by least squares using standard optimization algorithms.

They are very difficult to fit!

What is So Special About Nets?

With the exception of nets, all the methods that we shall discuss in the course are smoothers. At their core, they compute some measure of central tendency, usually a mean or a median:

> Least squares regression is just the generalization of the notion of a mean. A regression with only an intercept term actually is a mean.
>
> Nearest neighbors and decision trees carve up the space into cells and compute a mean at each point. That mean may be thresholded (majority vote) as we have done thus far, but need not be.

Neural nets can both **interpolate** and **smooth**. More importantly, they can decide themselves where in feature space interpolation is required and where smoothing is required.

To my knowledge, no other statistical procedure can do this.

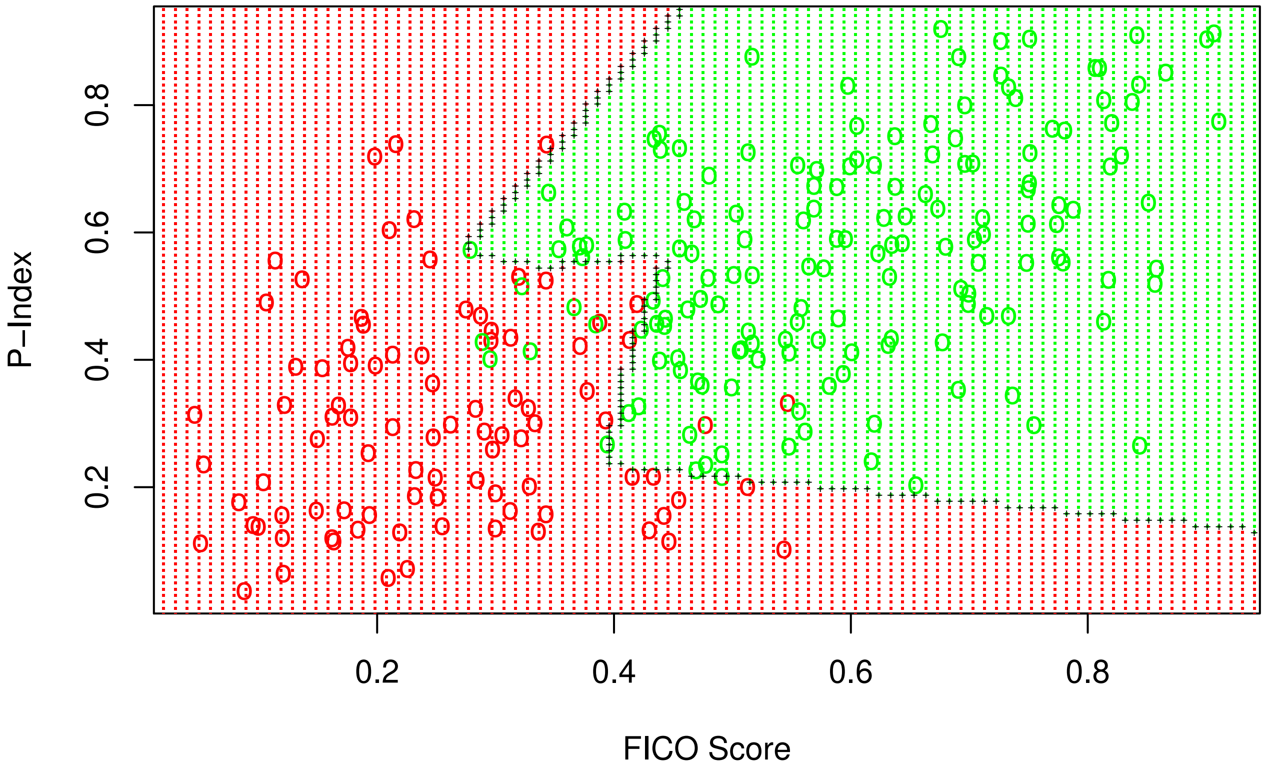
Unfortunately for us, this ability is more useful in robotics and chaotic dynamics than it is in data mining.

Neural Net Tuning

The tuning parameter for neural nets is the number of hidden units.

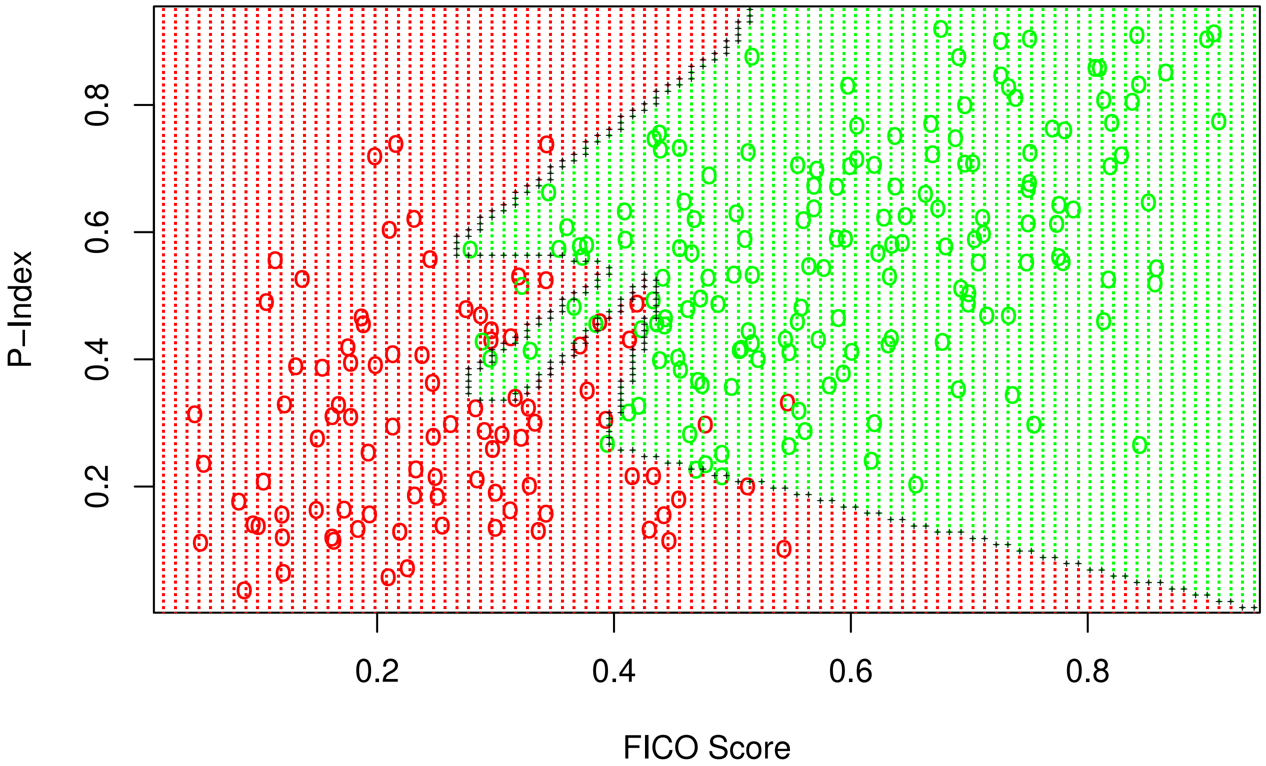
We shall try 5 and 8.

Fig 17. Neural Net, 5 Hidden Units



Shading indicates the classification regions; red predicts default and green predicts payment. The training set is shown as circles.

Fig 18. Neural Net, 8 Hidden Units



Shading indicates the classification regions; red predicts default and green predicts payment. The training set is shown as circles.

Table 3. Classification Error Rates

Method	Error Rate
Nearest Neighbor, $K = 15$	0.07697
Linear Regression	0.07854
Nearest Neighbor, $K = 5$	0.07943
Decision Tree, minsplit=20	0.08875
Decision Tree, minsplit=15	0.09149
Neural Net, 5 hidden units	0.09243
Neural Net, 8 hidden units	0.09698

Values within 0.002 can be regarded as the same.

Derived Features

We are not obliged to accept the features as they come. Features can be derived.

Common choices for derived features are logarithms and polynomials.

Logarithms are a good idea if we suspect that features act multiplicatively, i.e. if default were related to the product of x_1 and x_2. Polynomials are reasonable to try in general.

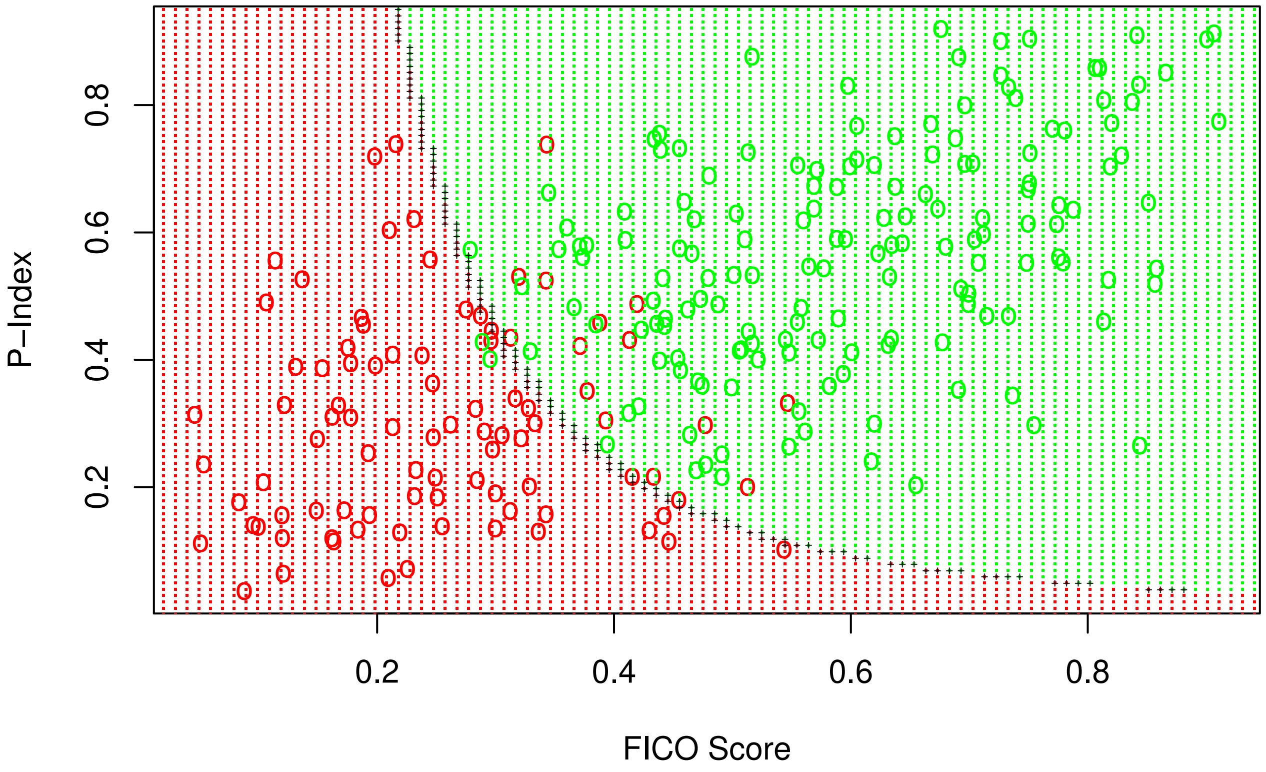
Let us see how regression works if we fit a model linear in logs

$$y = b_0 + b_1 \log(x_1) + b_2 \log(x_2)$$

and if we fit a quadratic

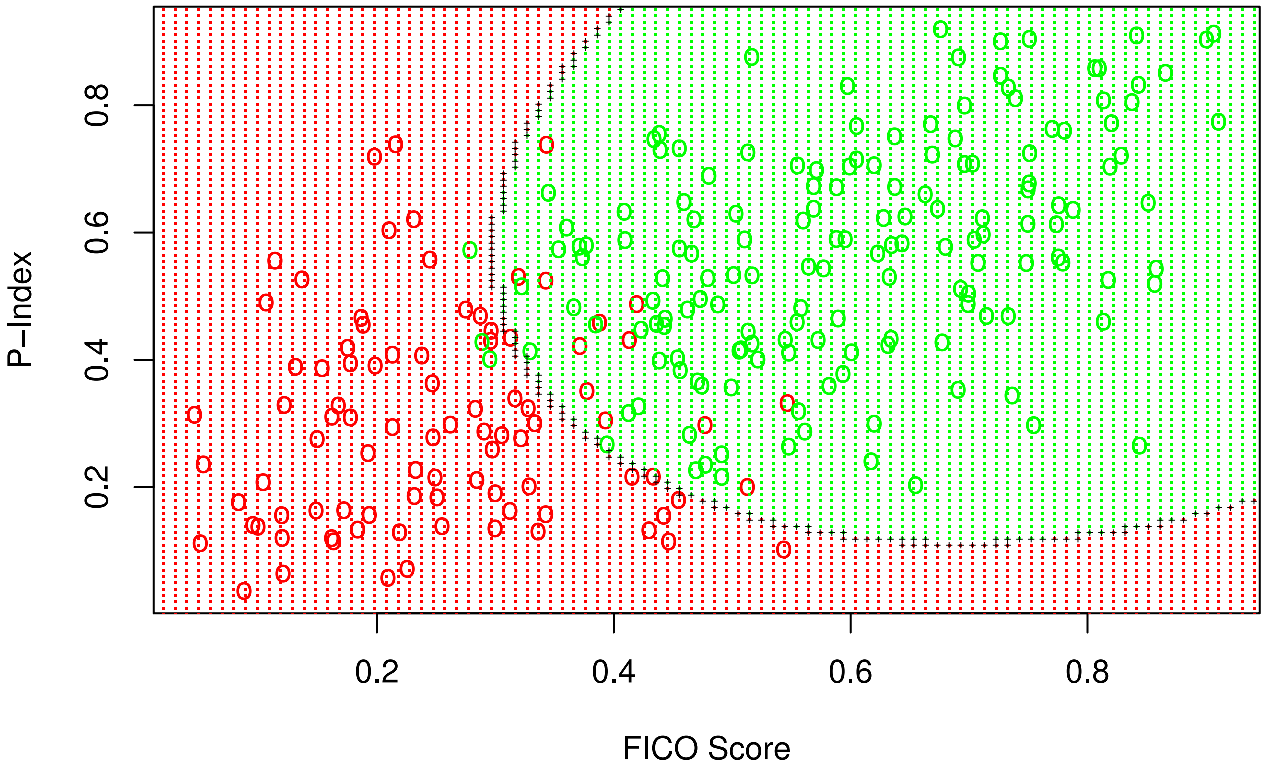
$$y = b_0 + b_1 x_1 + b_2 x_2 + b_3 (x_1)^2 + b_4 (x_2)^2 + b_5 (x_1)(x_2)$$

Fig 19. Log Linear Regression Classification



Shading indicates the classification regions; red predicts default and green predicts payment. The training set is shown as circles.

Fig 20. Quadratic Regression Classification



Shading indicates the classification regions; red predicts default and green predicts payment. The training set is shown as circles.

Table 4. Classification Error Rates

Method	Error Rate
Nearest Neighbor, $K = 15$	0.07697
Quadratic Regression	0.07807
Linear Regression	0.07854
Nearest Neighbor, $K = 5$	0.07943
Log Linear Regression	0.08394
Decision Tree, minsplit=20	0.08875
Decision Tree, minsplit=15	0.09149
Neural Net, 5 hidden units	0.09243
Neural Net, 8 hidden units	0.09698

Values within 0.002 can be regarded as the same.

An Important Point

What we learn from this is that if the features are chosen well, then even simple tools such as linear regression will work well.

Boosting

Boosting is the idea of using the scores from one tool as features for use by another tool. Berry and Linoff, pp. 213 – 221 describe four variants:

- **Segmented Input Combination Models** use different models for different parts of the input; only one model is used for a case.
- **Modeled Segmentation Combination Models** use the results from one model to segment the input, and then use another model to determine the output.
- **Error Fixing Combination Models** use high-confidence results from one model and build a separate model from the low confidence results.
- **Data Enhancement Combination Models** use the results of one model as input into another.

This is just the tip of the iceberg: Many other boosting methods have been proposed.

Boosting Regression

We shall boost the linear and quadratic regressions by using their scores (the $\widehat{y}s$) as the input to a regression tree. This is a data enhancement combination model method.

A two leaf tree improves results. Results deteriorate when the tree is more complicated. Here are the revised classification rules:

- Linear Regression: 1 if $\widehat{y}_i \geq 0.5248545$, 0 else

- Quadratic Regression: 1 if $\widehat{y}_i \geq 0.4294637$, 0 else

The next table shows how we do.

Table 5. Classification Error Rates

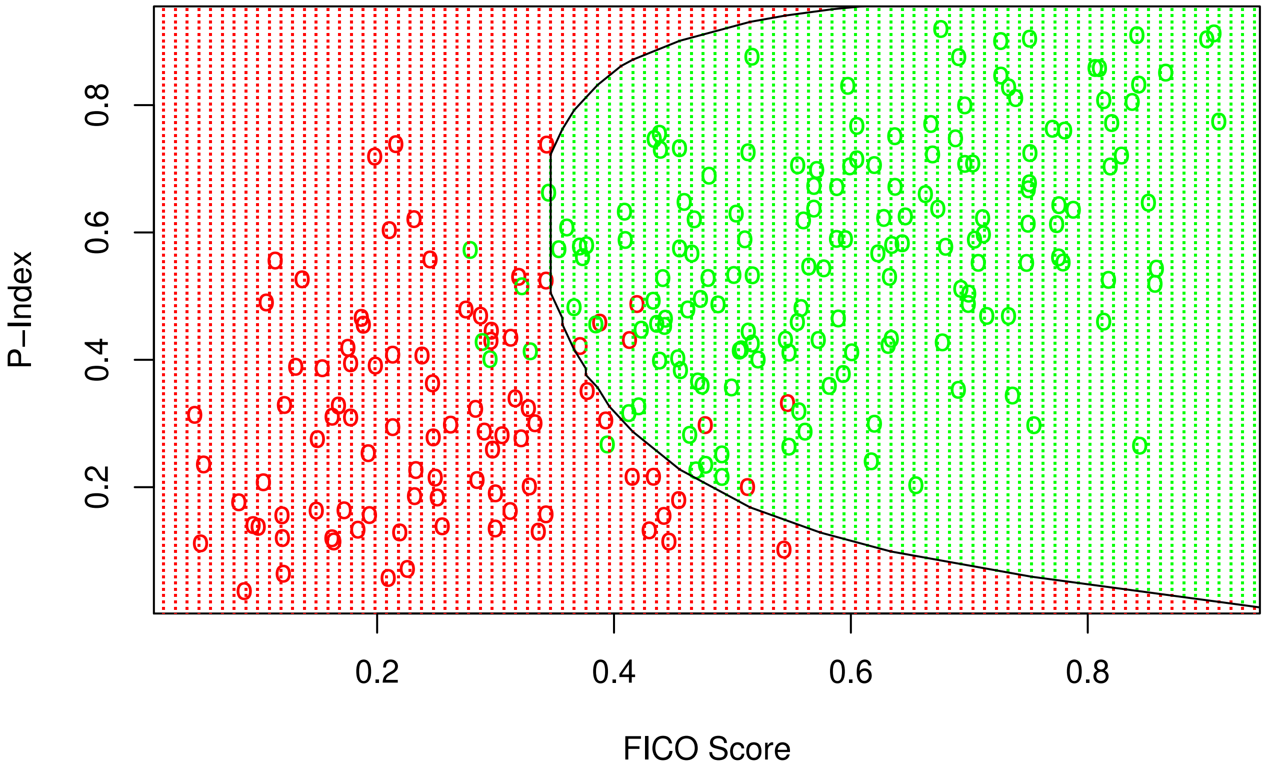
Method	Error Rate
Boosted Quadratic Regression	0.07493
Boosted Linear Regression	0.07498
Nearest Neighbor, $K = 15$	0.07697
Quadratic Regression	0.07807
Linear Regression	0.07854
Nearest Neighbor, $K = 5$	0.07943
Log Linear Regression	0.08394
Decision Tree, minsplit=20	0.08875
Decision Tree, minsplit=15	0.09149
Neural Net, 5 hidden units	0.09243
Neural Net, 8 hidden units	0.09698

Values within 0.002 can be regarded as the same.

How Well Have We Done

Because I know the true model, I can derive the theoretically optimal classification rule. Here it is:

Fig 21. The Optimal Classification



Shading indicates the classification regions; red predicts default and green predicts payment. The training set is shown as circles.

Table 6. Classification Error Rates

Method	Error Rate
Optimal	0.07397
Boosted Quadratic Regression	0.07493
Boosted Linear Regression	0.07498
Nearest Neighbor, $K = 15$	0.07697
Quadratic Regression	0.07807
Linear Regression	0.07854
Nearest Neighbor, $K = 5$	0.07943
Log Linear Regression	0.08394
Decision Tree, minsplit=20	0.08875
Decision Tree, minsplit=15	0.09149
Neural Net, 5 hidden units	0.09243
Neural Net, 8 hidden units	0.09698

Values within 0.002 can be regarded as the same.

Decision Forests

A form of boosting for decision trees.

Take a random sample from the data of about 10% and fit a bushy tree. Replace the sample and repeat about 25 times.

A prediction for a given case is the average of the predictions of the 25 trees.

Used by Xiaoxi Li, Principal Global Investors, Class of 2006, for portfolio selection.

Dictionary Methods

Dictionary models have the form

$$
\begin{aligned}
y &= \beta_0 + \beta_{11} f_{11}(x_1) + \cdots + \beta_{1d} f_{1d}(x_p) \\
&\quad + \beta_{21} f_{21}(x_1, x_2) + \cdots + \beta_{2d_2} f_{1d_2}(x_{p-1}, x_p) \\
&\quad + \cdots + \beta_{dd_d} f_{1d_d}(x_1, \ldots, x_{p-1}, x_p)
\end{aligned}
$$

where f_{ij} are derived features chosen from a "dictionary".

The examples we have seen so far are linear regression, where the dictionary items were x_1 and x_2, regression with derived features, where the dictionary items were $\log(x_1)$ and $\log(x_2)$ or x_1, x_2, x_1^2, x_2^2, and $x_1 x_1$, and neural nets, where the dictionary items were

$$
S\Big(\beta_{0j} + \beta_{1j} x_{1j} + \beta_{2j} x_{2j}\Big),
$$

where S is the "squasher".

Dictionary Methods

Most data mining tools are dictionary models; e.g. regression, neural nets, projection pursuit, support vector machines, radial basis functions, wavelets, thin-plate splines.

Some, like regression, rely on the user to make an intelligent choice of derived features.

Others, like neural nets, try to automate the choice of derived features.

Obviously a blend works the best. Use intelligence to find good derived features to your best ability and then use the machine to find what features it can.

Comments on Nearest Neighbors

What we have just seen is true in general: Nearest neighbors will give nearly optimal answers in large data sets that have a small number of features.

The problem is the caveat "that have a small number of features." Nearest neighbors is a localization method: The value it predicts at a point $x^o = (x_1^o, x_2^o)$ depends only on the cases $x_i = (x_{1i}, x_{2i})$ in the training data that are close to x^o. Some other localization methods are kernel regression, local linear regression, and local quadratic regression. All localization methods suffer from the fact that they will only work well when the number of features is small. This problem is often called the "curse of dimensionality."

The Curse of Dimensionality

The problem faced by localization methods can be seen from the mathematical formula for the nearest neighbor prediction

$$y(x^o) = \frac{1}{K} \sum_{x_i \in \mathcal{N}_K(x^o)} y_i$$

As seen from the formula, the size of neighborhood $\mathcal{N}_K(x^o)$ must be small so that the neighboring cases $x_i \in \mathcal{N}_k(x^o)$ from the training set are truly representative of x^o. Also, K must be of a reasonable size so that the average of the corresponding targets y_i is stable.

Supposing that x^o has p features, i.e. $x^o = (x_1^o, \ldots x_p^o)$, we see that for the method to work well we must be able to populate the training data with enough cases $x_i = (x_{i1}, x_{i2}, \ldots, x_{ip})$ so that several of them have their feature x_{i1} near enough to x_1^o; similarly for $x_{i2}, \ldots, x_{ip}$.

The Curse of Dimensionality

Suppose that the unit of measurement of each coordinate is length and the relevant range is a foot.

In one dimension, we can have features that are no more than one inch apart by dividing the foot into inches and putting a point at the center of each interval. This is 12 cases.

In two dimensions, we can have features that are no more than one inch apart in each coordinate by dividing the square foot into square inches and putting a point at the center of each. This is 144 cases.

In three dimensions, divide the cubic foot into cubic inches and put one point at each center. This is 1,728 cases.

In ten dimensions this is $12^{10} = 61,917,364,224$ cases!

Comments on Dictionary Methods

The best of the dictionary models are neural nets. They can achieve the performance of the nearest neighbor method without suffering unduly from the curse of dimensionality.

The downside is that they are quirky and very hard to fit. The fit shown as Figure 17 is far from the best achievable. We shall discuss computational strategies later. As we shall see, effective algorithms are not cheap: Determining a well fitting net is a computationally intensive task.

Comments on Decision Trees

Classification and regression trees are another approach to combating the curse of dimensionality. They try to partition feature space in such a way that simple models can be used over subsets of feature space.

Aside from working well, decision trees have two strong advantages: They are cheaper to compute than many competitors and they partition feature space into what can be meaningful subsets.

To some extent, decisions trees simultaneously accomplish both supervised learning and unsupervised learning. They can be viewed as a single-shot method.

The Main Points

1. Tools

2. Validation

3. Derived Features

Tools

You have been introduced to the main tools of data mining: regression, logistic regression, nearest neighbors, decision trees, and neural nets.

The main omissions are discriminant analysis, principal components, hierarchical clustering, K-means.

Discriminant analysis is needed to have a practical tool for categorical data with more than two categories. Logistic regression is needed because everyone will expect you to know it.

The other three are the main tools of unsupervised learning.

You understand the limitation of localization methods: The curse of dimensionality.

Validation

Validation is the central idea of data mining.

The attitude that models are to be tuned and selected by means of a hold out sample or by cross validation and not by within sample statistical testing pervades data mining no matter who does it: statistician, computer scientist, engineer, MBA, etc.

Usually data sets are large so validation is used rather than cross validation.

Derived Features

What one is really trying to do in data mining is find the best (derived) features in the data.

If the features are well chosen, then even simple linear tools like regression will work well.

Blank page

Blank page