Supervised learning - I

M. Vazirgiannis

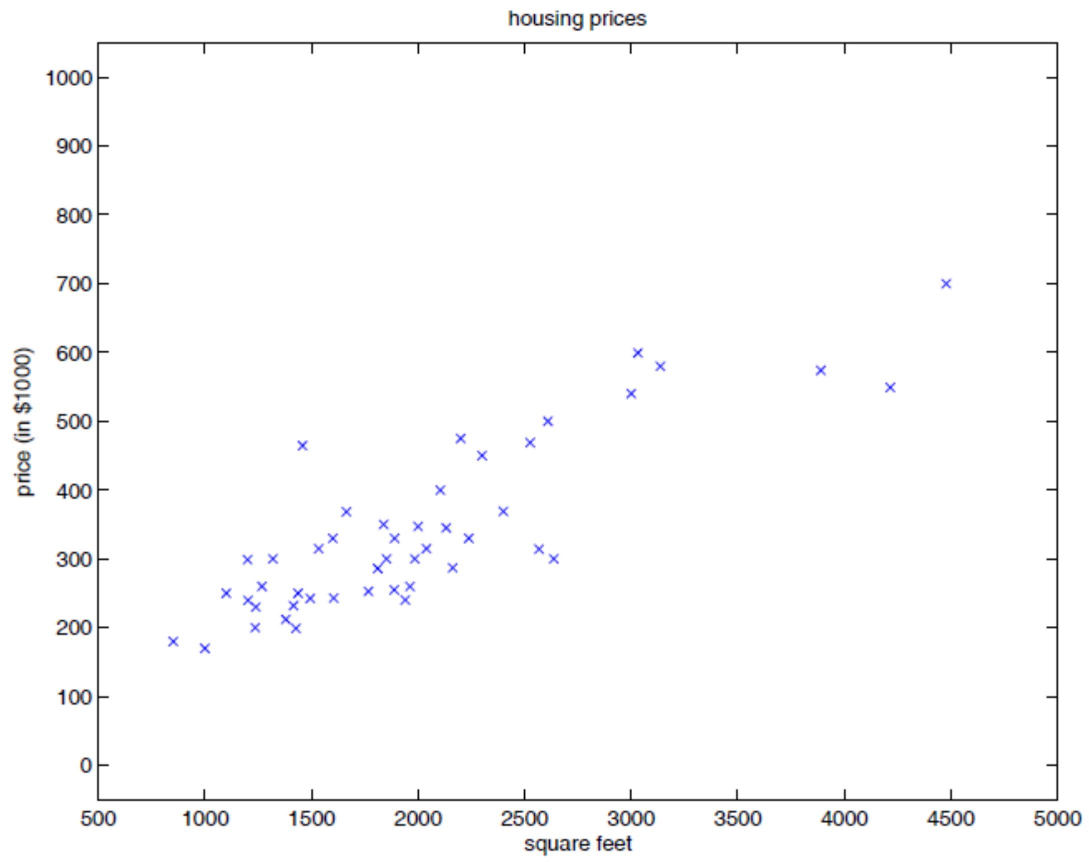
December 2014

Outline

- **Introduction to supervised learning**
- Regression
- Naïve Bayes
- K-nn
- Decision Trees

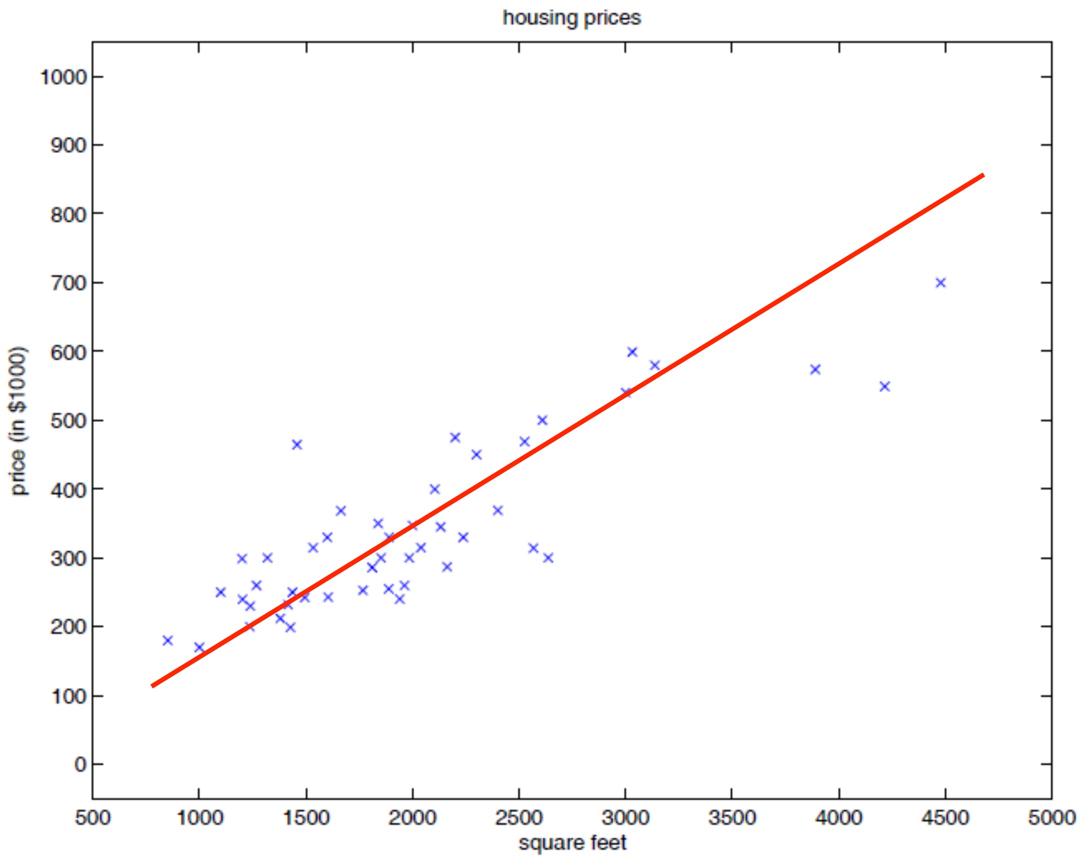
Prediction..

$\{(x_1, y_1), \; ..$



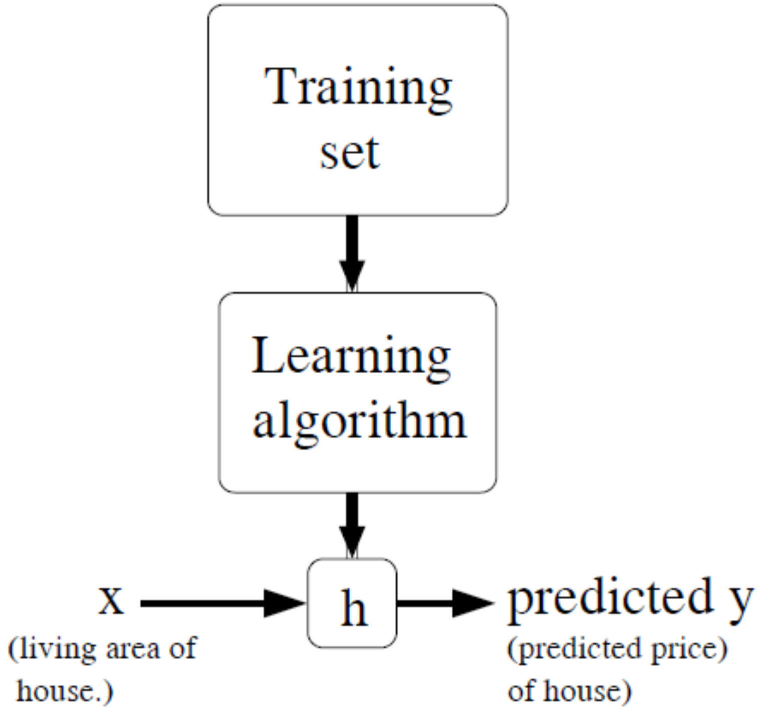
- Can we predict the price of a house based on its size (suface in m^2) ?

Prediction..



housing prices
price (in $1000)
square feet
0
100
200
300
400
500
600
700
800
900
1000
500
1000
1500
2000
2500
3000
3500
4000
4500
5000

Prediction..



- y continuous value: ***prediction***
- y discrete value: ***classification***

Prediction..

- x(i): “input” variables (input features)
- y(i): “output” or target variable that we are trying to predict
- A pair (x(i), y(i)) is called a training example,
- The *training set* - a list of m training examples

  {(x(i), y(i)); i =1, . . . ,m}—is called a training set.
- X : space of input values, Y : output values.
- The supervised learning problem:
  - given a training set,
  - learn a function **h : X → Y** so that **h(x)** is a “good” predictor for the corresponding value of **y**.

  For historical reasons, this function **h** is called a hypothesis.

Classes of classifiers

- Class-conditional/probabilistic, based on $p(\underline{x} \mid c_k)$,
  - Naïve Bayes: simple, but often effective in high dimensions
  - Parametric generative models, e.g., Gaussian (can be effective in low-dimensional problems: leads to quadratic boundaries in general)

- Regression-based, $p(c_k \mid \underline{x})$ directly
  - Logistic regression: simple
  - Neural network: non-linear extension of logistic, can be difficult to work with

Classes of classifiers

- Discriminative models, focus on locating optimal decision boundaries
  - Linear discriminants, perceptron: simple, sometimes effective
  - Support vector machines: generalization of linear discriminants, can be quite effective, computational complexity is an issue
  - Nearest neighbor: simple, can scale poorly in high dimensions
  - Decision trees: "swiss army knife", often effective in high dimensionis

Results Evaluation metrics

- Confusion matrix:

	Actual class	
Predicted class	True Positive	False Positive
	False Negative	True Negative

- Precision $\frac{TP}{TP + FP}$
- Recall $\frac{TP}{TP + FN}$
- Matching coefficient $\frac{TP + TN}{TP + TN + FP + FN}$

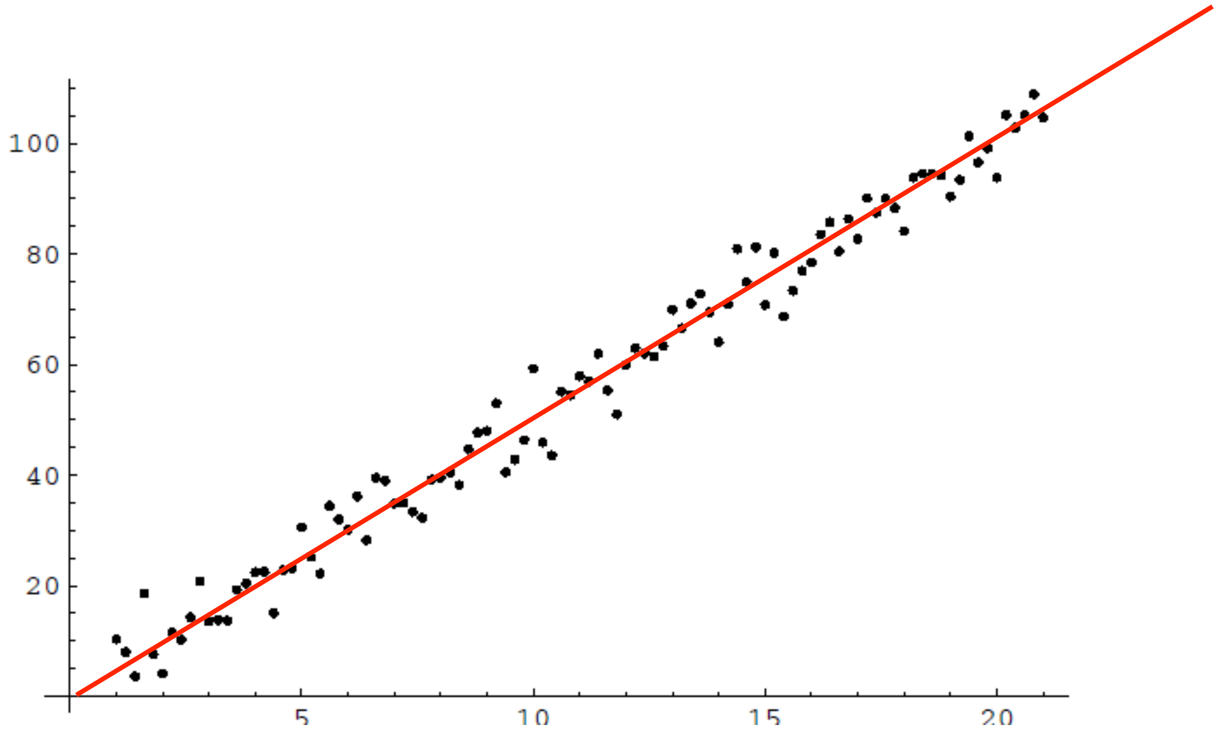
Outline

- Introduction to supervised learning
- **Regression**
- Naïve Bayes
- K-nn
- Decision Trees

Regression

- Aims at fitting a line to a set of observations

$\{(x_1, y_1), \ldots, (x_N, y_N)\}$, there is a straight line $y = ax + b$.



Regression

- individual point error is: $y-(ax+b)$
- thus the error set is: $\{y_1 - (ax_1 + b),\ \ldots,\ y_N - (ax_N + b)\}.$
- the total error is: $E(a,b) = \sum_{n=1}^{N} (y_n - (ax_n + b))^2 .$

known as SSE as well

Least Squares method

- Objective is to minimize $E(a,b) = \sum_{n=1}^{N} (y_n - (ax_n + b))^2 .$

- Thus to find values α, b such that: $\frac{\partial E}{\partial a} = 0, \quad \frac{\partial E}{\partial b} = 0.$

- Differentiation leads to:

$$\begin{aligned} \frac{\partial E}{\partial a} &= \sum_{n=1}^{N} 2\,(y_n - (ax_n + b)) \cdot (-x_n) \\ \frac{\partial E}{\partial b} &= \sum_{n=1}^{N} 2\,(y_n - (ax_n + b)) \cdot 1. \end{aligned}$$

Least Squares method

- Thus setting $$\frac{\partial E}{\partial a} = 0, \quad \frac{\partial E}{\partial b} = 0.$$
- Leads to:

$$\begin{aligned}\sum_{n=1}^{N} (y_n - (ax_n + b)) \cdot x_n &= 0 \\ \sum_{n=1}^{N} (y_n - (ax_n + b)) &= 0.\end{aligned}$$

- Or equivalently:

$$\begin{aligned}\left(\sum_{n=1}^{N} x_n^2\right) a + \left(\sum_{n=1}^{N} x_n\right) b &= \sum_{n=1}^{N} x_n y_n \\ \left(\sum_{n=1}^{N} x_n\right) a + \left(\sum_{n=1}^{N} 1\right) b &= \sum_{n=1}^{N} y_n.\end{aligned}$$

Least Squares method

- Hence

$$\begin{pmatrix} \sum_{n=1}^{N} x_n^2 & \sum_{n=1}^{N} x_n \\ \sum_{n=1}^{N} x_n & \sum_{n=1}^{N} 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sum_{n=1}^{N} x_n y_n \\ \sum_{n=1}^{N} y_n \end{pmatrix}$$

- Implying:

$$\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sum_{n=1}^{N} x_n^2 & \sum_{n=1}^{N} x_n \\ \sum_{n=1}^{N} x_n & \sum_{n=1}^{N} 1 \end{pmatrix}^{-1} \begin{pmatrix} \sum_{n=1}^{N} x_n y_n \\ \sum_{n=1}^{N} y_n \end{pmatrix}$$

Least Squares method -

$$\begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sum_{n=1}^{N} x_n^2 & \sum_{n=1}^{N} x_n \\ \sum_{n=1}^{N} x_n & \sum_{n=1}^{N} 1 \end{pmatrix}^{-1} \begin{pmatrix} \sum_{n=1}^{N} x_n y_n \\ \sum_{n=1}^{N} y_n \end{pmatrix}$$

Solution only if $\begin{pmatrix} \sum_{n=1}^{N} x_n^2 & \sum_{n=1}^{N} x_n \\ \sum_{n=1}^{N} x_n & \sum_{n=1}^{N} 1 \end{pmatrix}$ is invertible

- i.e. if it determinant is **not** 0

Least Squares method -

The method is generalized in a straight forward way:

Assume $y = af(x) + bg(x)$ then the respective result is:

$$\begin{pmatrix} \sum_{n=1}^{N} f(x_n)^2 & \sum_{n=1}^{N} f(x_n)g(x_n) \\ \sum_{n=1}^{N} f(x_n)g(x_n) & \sum_{n=1}^{N} g(x_n)^2 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sum_{n=1}^{N} f(x_n)y_n \\ \sum_{n=1}^{N} g(x_n)y_n \end{pmatrix}$$

Outline

- Introduction to supervised learning
- Regression
- **Naïve Bayes**
- K-nn
- Decision Trees

Bayesian Classification: Why?

- <u>Probabilistic learning</u>
  - Calculate explicit probabilities for hypothesis,
  - practical approaches to certain types of learning problems
- <u>Incremental</u>
  - Each training example can incrementally increase/decrease the probability that a hypothesis is correct.
  - Prior knowledge combined with observed data.
- <u>Probabilistic prediction</u>
  - Predict multiple hypotheses, weighted by their probabilities

Bayesian classification

- The classification problem may be formalized using a-posteriori probabilities:
- $P(C|X)$ = prob. that the sample tuple $X=<x_1,\ldots,x_k>$ is of class C.

- e.g. P(class=N | outlook=sunny,windy=true,…)

- Idea: assign to sample X the class label C such that $P(C|X)$ is maximal

Estimating a-posteriori probabilities

- Bayes theorem:

$$P(C|X) = P(X|C) \cdot P(C) / P(X)$$

- $P(X)$ is constant for all classes
- $P(C)$ = relative freq of class C samples
- C such that $P(C|X)$ is maximum =
  C such that $P(X|C) \cdot P(C)$ is maximum
- Problem: computing $P(X|C)$ is unfeasible!

Naïve Bayesian Classification

- Naïve assumption: attribute independence

  $$P(x_1,\dots,x_k|C) = P(x_1|C)\cdot\ldots\cdot P(x_k|C)$$

- i-th attribute is categorical:
  $P(x_i|C)$: relative frequency of samples having value x_i as i-th attribute in class C
- If i-th attribute is continuous:
  - Real-valued variables discretized to create nominal versions
  - $P(x_i|C)$ is estimated thru a Gaussian density function
- Computationally feasible in both cases
- Generative probabilistic model with conditional independence assumption on $p(\underline{x} \mid c_k)$, i.e.

  $$p(\underline{x} \mid c_k) = \prod p(x_j \mid c_k)$$

Naïve Bayes Classifiers

- Comments:
  - Simple to train
    - estimate conditional probabilities for each feature-class pair
  - Often works surprisingly well in practice
    - e.g., state of the art for text-classification, basis of many widely used spam filters
  - Feature selection can be helpful, e.g., information gain
  - Note that even if CI assumptions are not met, it may still be able to approximate the optimal decision boundaries (seems to happen in practice)
  - However…. on most problems can usually be beaten with a more complex model (plus more work)

Play-tennis example: estimating $P(x_i|C)$

Outlook	Temperature	Humidity	Windy	Class
sunny	hot	high	false	N
sunny	hot	high	true	N
overcast	hot	high	false	P
rain	mild	high	false	P
rain	cool	normal	false	P
rain	cool	normal	true	N
overcast	cool	normal	true	P
sunny	mild	high	false	N
sunny	cool	normal	false	P
rain	mild	normal	false	P
sunny	mild	normal	true	P
overcast	mild	high	true	P
overcast	hot	normal	false	P
rain	mild	high	true	N

P(p) = 9/14
P(n) = 5/14

outlook	
P(sunny|p) = 2/9	**P(sunny|n) = 3/5**
P(overcast|p) = 4/9	**P(overcast|n) = 0**
P(rain|p) = 3/9	**P(rain|n) = 2/5**
temperature	
P(hot|p) = 2/9	**P(hot|n) = 2/5**
P(mild|p) = 4/9	**P(mild|n) = 2/5**
P(cool|p) = 3/9	**P(cool|n) = 1/5**
humidity	
P(high|p) = 3/9	**P(high|n) = 4/5**
P(normal|p) = 6/9	**P(normal|n) = 2/5**
windy	
P(true|p) = 3/9	**P(true|n) = 3/5**
P(false|p) = 6/9	**P(false|n) = 2/5**

Play-tennis example: classifying X

- An unseen sample X = <rain, hot, high, false>

- P(X|p)·P(p) =
  P(rain|p)·P(hot|p)·P(high|p)·P(false|p)·P(p) =
  3/9·2/9·3/9·6/9·9/14 = 0.010582
- P(X|n)·P(n) =
  P(rain|n)·P(hot|n)·P(high|n)·P(false|n)·P(n) =
  2/5·2/5·4/5·2/5·5/14 = 0.018286
- Sample X is classified in class n (don't play)

The independence hypothesis…

- makes computation possible
- yields optimal classifiers when satisfied
- but is seldom satisfied in practice, as attributes (variables) are often correlated.
- Attempts to overcome this limitation:
  - Bayesian networks, that combine Bayesian reasoning with causal relationships between attributes
  - Decision trees, that reason on one attribute at the time, considering most important attributes first

Outline

- Introduction to supervised learning
- Regression
- Naïve Bayes
- **K-nn**
- Decision Trees

Nearest Neighbor Classifiers

- K-nn: select the k nearest neighbors to x from the training data and select the majority class from these neighbors
- k is a parameter:
  - Small k: "noisier" estimates, Large k: "smoother" estimates
  - Best value of k often chosen by cross-validation
- Comments
  - Virtually assumption free
  - Interesting theoretical properties:

    Bayes error < error(kNN) < 2 x Bayes error (asymptotically)
- Disadvantages
  - Can scale poorly with dimensionality: sensitive to distance metric
  - Requires fast lookup at run-time to do classification with large n
  - Does not provide any interpretable "model"

Outline

- Introduction to supervised learning

- Regression
- Naïve Bayes
- K-nn
- **Decision Trees**

Decision Tree Classifiers

- Widely used in practice
  - Can handle both real-valued and nominal inputs (unusual)
  - Good with high-dimensional data

– similar algorithms as used in constructing regression trees

– historically, developed both in statistics and computer science
  - Statistics:
    - Breiman, Friedman, Olshen and Stone, CART, 1984
  - Computer science:
    - Quinlan, ID3, C4.5 (1980's-1990's)

Entropy & Information gain

- **Entropy**: measures the randomness of a statistical variable (or otherwise the information the variable carries)

$$H(x) = \sum_{i=1}^{n} p(i) \log_2 \left(\frac{1}{p(i)} \right) = -\sum_{i=1}^{n} p(i) \log_2 p(i).$$

- **Information gain** is the change in entropy from a prior state to a state that takes some information as given:

$$IG(Ex,a) = H(Ex) - H(Ex \mid a)$$

Information Gain (ID3/C4.5)

- Select the attribute with the highest information gain
- Assume there are two classes, *P* and *N*
  - Let the set of examples *S* contain *p* elements of class *P* and *n* elements of class *N*
  - The amount of information, needed to decide if an arbitrary example in *S* belongs to *P* or *N* is defined as

$$I(p,n) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

Information Gain in Decision Tree Induction

- Assume that using attribute A a set *S* will be partitioned into sets $\{S_1, S_2, \ldots, S_v\}$
  - If S_i contains p_i examples of *P* and n_i examples of *N*, the entropy, or the expected information needed to classify objects in all subtrees S_i is

$$E(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p + n} I(p_i, n_i)$$

- The encoding information that would be gained by branching on *A*

$$Gain(A) = I(p,n) - E(A)$$

Attribute Selection by Information Gain Computation

- Class P: buys_computer = "yes"
- Class N: buys_computer = "no"
- $I(p, n) = I(9, 5) = 0.940$
- Compute the entropy for *age*:

age	p_i	n_i	$I(p_i, n_i)$
<=30	2	3	0,971
30…40	4	0	0
>40	3	2	0,971

$$E(age) = \frac{5}{14} I(2,3) + \frac{4}{14} I(4,0) + \frac{5}{14} I(3,2) = 0.69$$

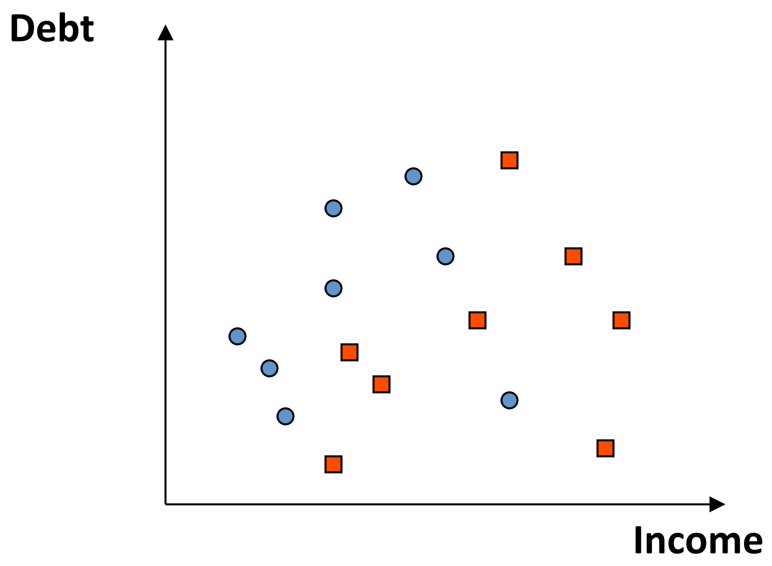
Hence

$$Gain(age) = I(p,n) - E(age)$$

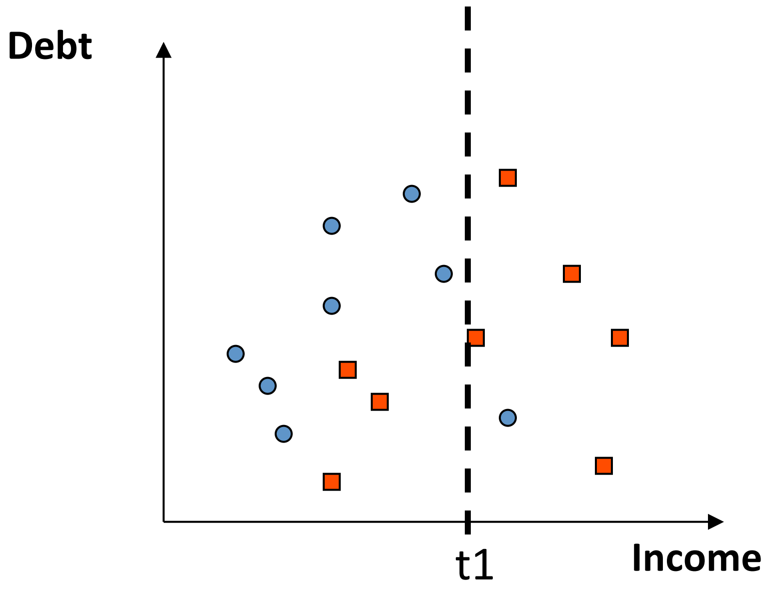
Similarly

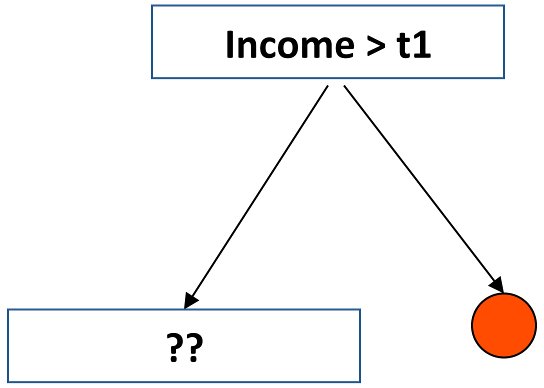
$$Gain(income) = 0.029$$
$$Gain(student) = 0.151$$
$$Gain(credit\_rating) = 0.048$$

Decision Tree Example

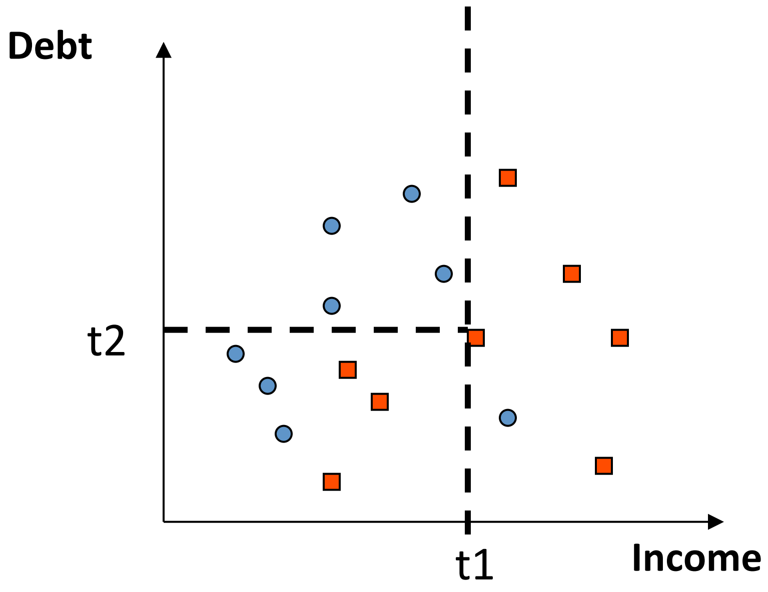


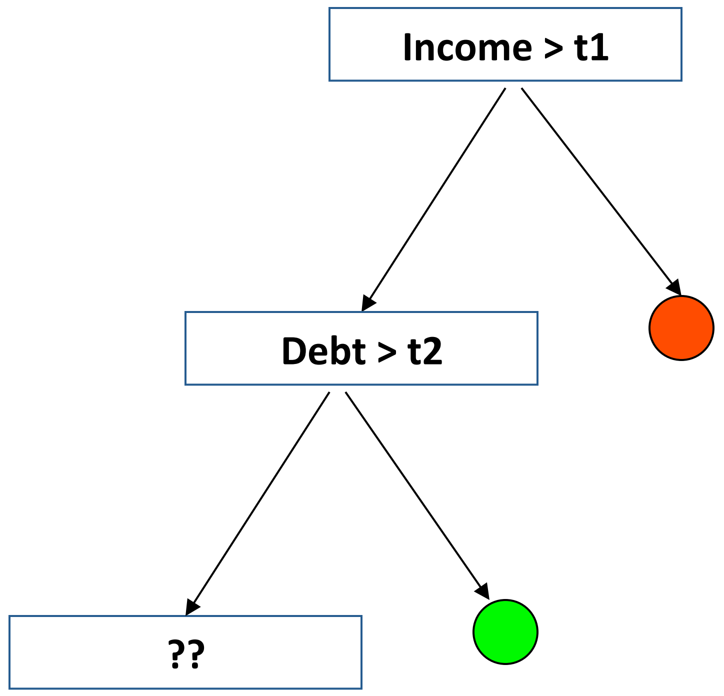
Decision Tree Example



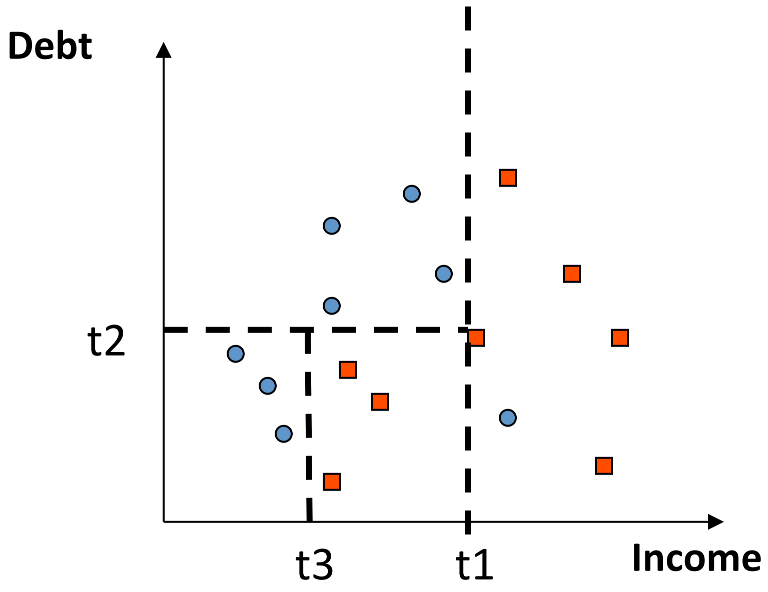


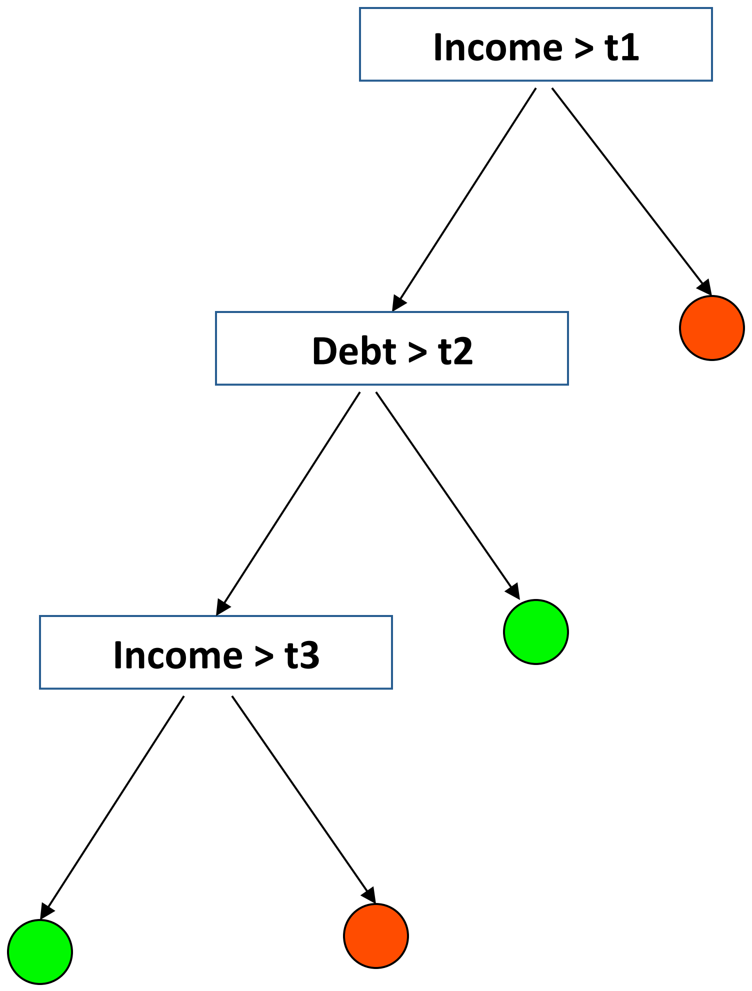
Decision Tree Example



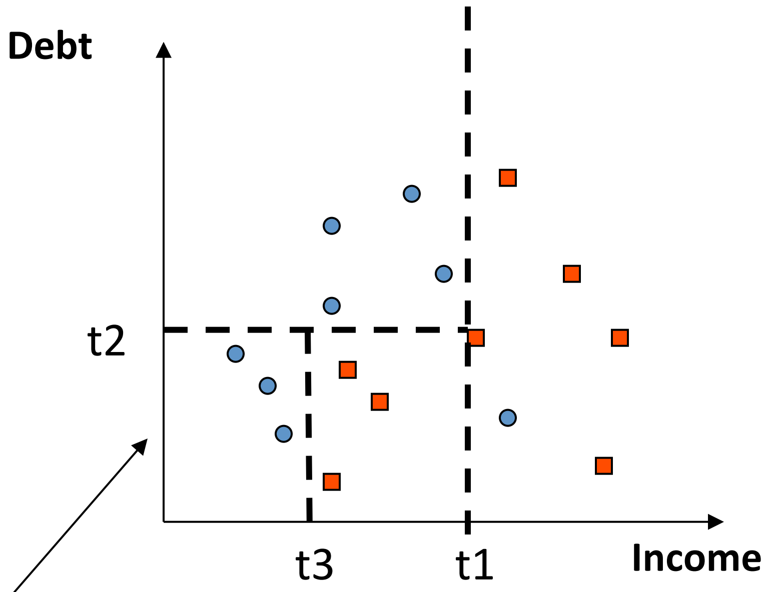


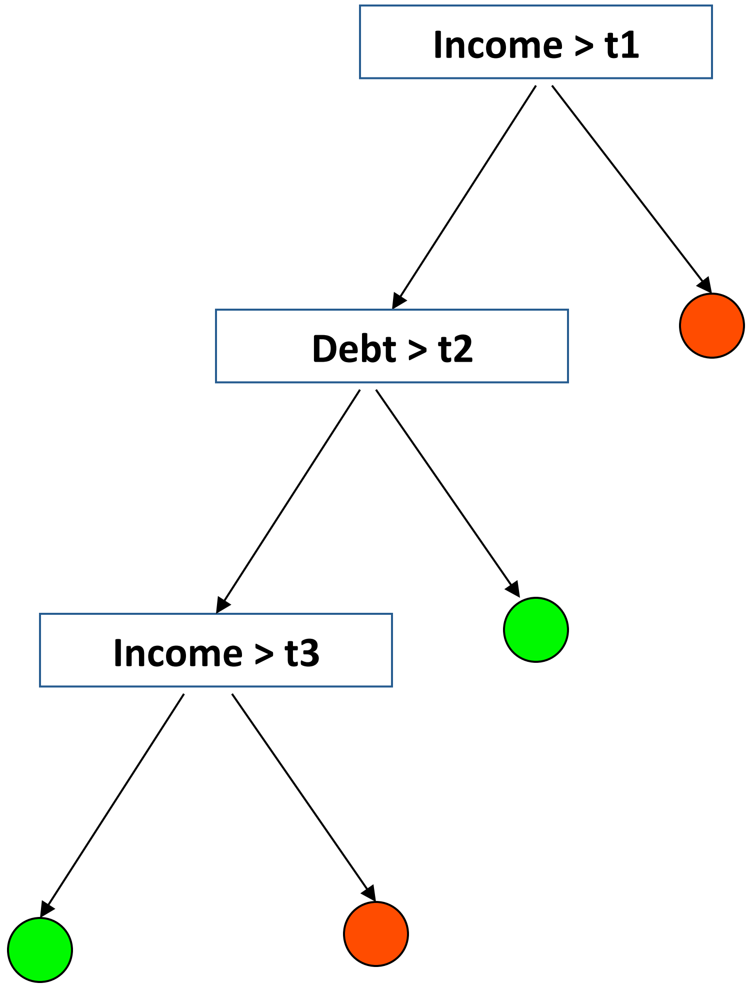
Decision Tree Example





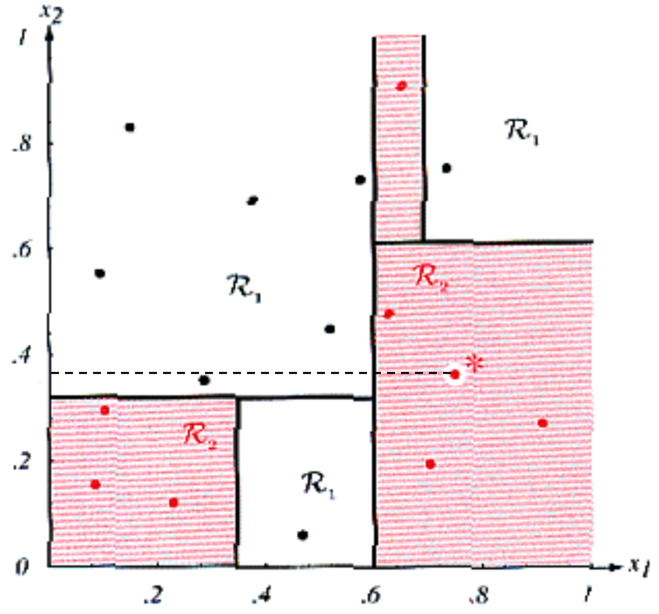
Decision Tree Example

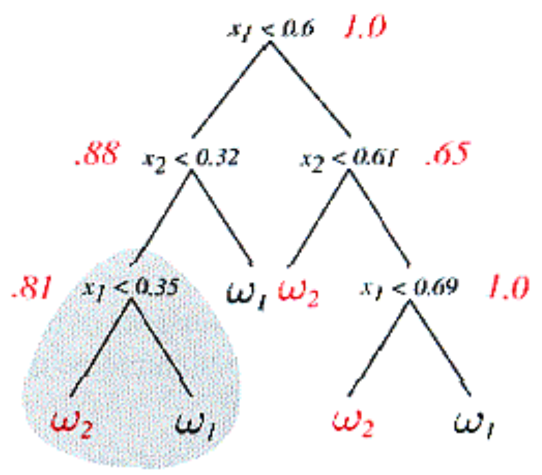




Note: tree boundaries are piecewise linear and axis-parallel

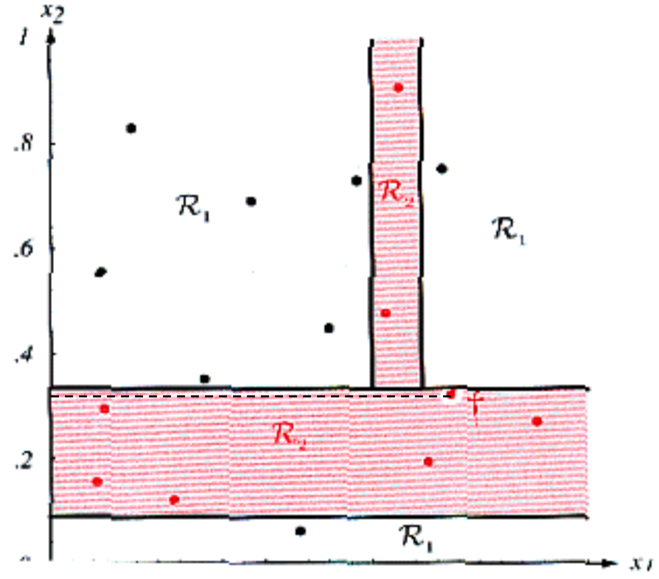
Decision Trees are not stable





Moving just one example slightly may lead to quite different trees and space partition!

Lack of stability against small perturbation of data.



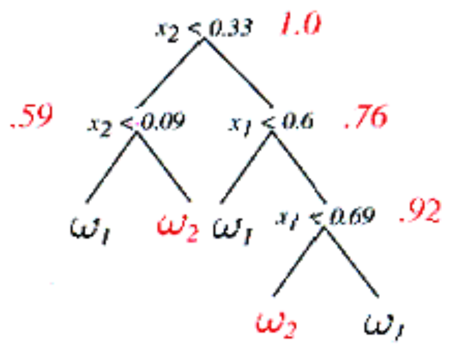


Figure from
Duda, Hart & Stork,
Chap. 8

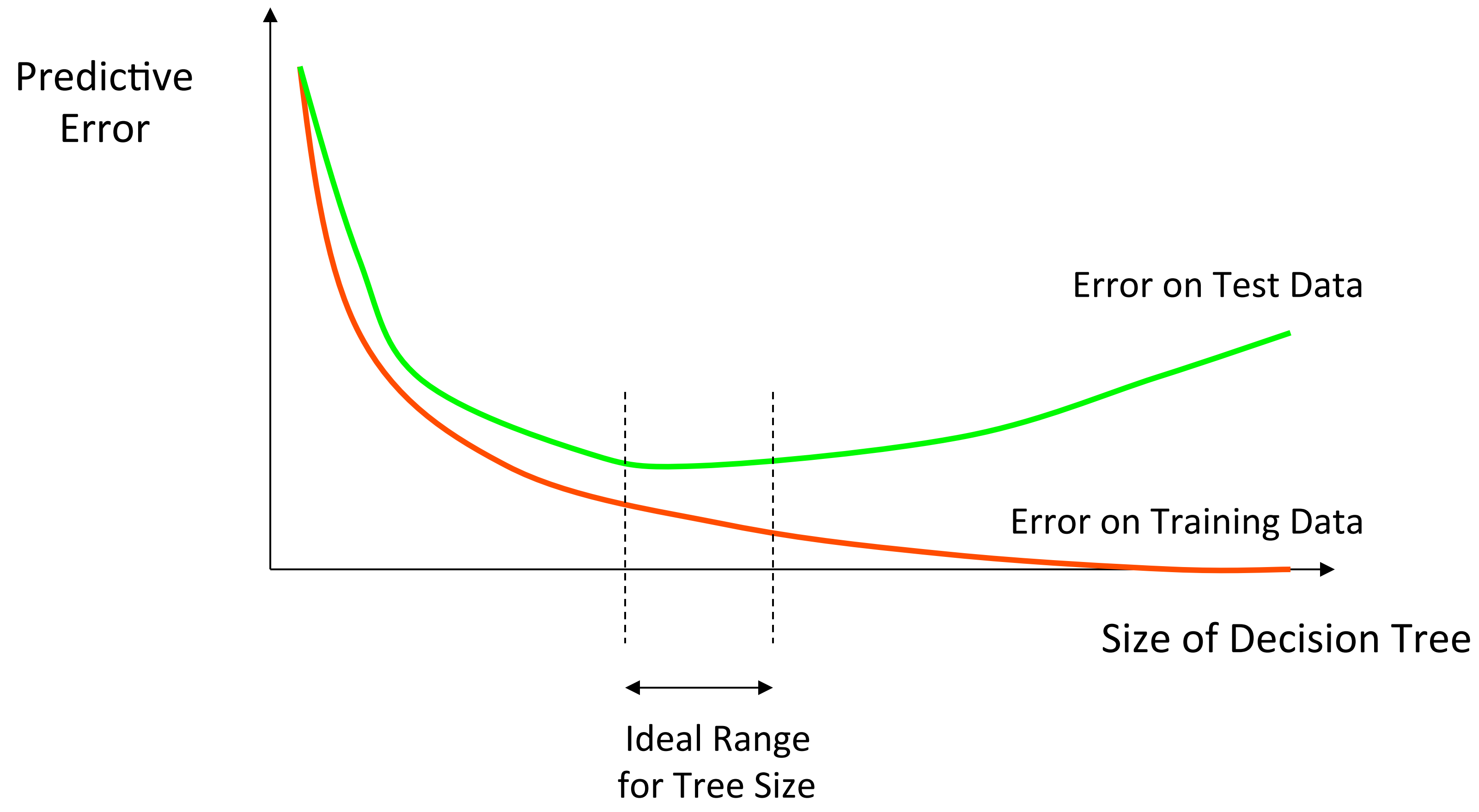
Splitting on a nominal attribute

- Nominal attribute with m values
  - e.g., the name of a state or a city in marketing data

- 2^{m-1} possible subsets => exhaustive search is $O(2^{m-1})$
  - For small m, a simple approach is to branch on specific values
  - But for large m this may not work well

- Neat trick for the 2-class problem:
  - For each predictor value calculate the proportion of class 1's
  - Order the m values according to these proportions
  - Now treat as an ordinal variable and select the best split (linear in m)
  - This gives the optimal split for the Gini index, among all possible 2^{m-1} splits (Breiman et al, 1984).

How to Choose the Right-Sized Tree?



Why Trees are widely used in Practice

- Can handle high dimensional data
  - builds a model using 1 dimension at time
- Can handle any type of input variables
  - categorical, real-valued, etc
  - most other methods require data of a single type (e.g., only real-valued)
- Trees are (somewhat) interpretable
  - domain expert can "read off" the tree's logic
- Tree algorithms are relatively easy to code and test

Limitations of Trees

- Representational Bias
  - classification: piecewise linear boundaries, parallel to axes
  - regression: piecewise constant surfaces

- High Variance
  - trees can be "unstable" as a function of the sample
    - e.g., small change in the data -> completely different tree
  - causes two problems
    - High variance contributes to prediction error
    - High variance reduces interpretability
  - Trees are good candidates for model combining
    - Often used with boosting and bagging

- Trees do not scale well to massive data sets (e.g., N in millions)
  - repeated random access of subsets of the data