AiFo ZF HS 21

Contents

1. Natural Language Processing (Word Embeddings)

Words need to be stored as numbers to work with them in an AI environment.

One-Hot Representation: Count number of different words, represent each unique word with a one-hot vector:

It	rains	.	We	go	home	.	It	stops	raining	.
1	0	0	0	0	0	0	1	0	0	0
0	1	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	1
0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	0	0	0	0	0	1	0

Results in a very high dimensional vector space. Very sparse vectors. No meaning.

Indexing: Make a list of unique words. Index them and use this index to refer to them.

cat	→ 0
mat	→ 1
on	→ 2
sat	→ 3
the	→ 4
.	→ 5

"The cat sat on the mat" → [4, 0, 3, 2, 4, 1, 5]

Dense equivalent to one-hot encoding. No Meaning.

Distributed Representation (Dense Vectors): You shall know a word by the company it keeps. Words that appear in similar contexts tend to have similar meaning.

Words are transformed to high dimensional vectors that incorporate the meaning of the word by looking at surrounding words. The resulting vectors are similar for similar words. **Cosine similarity** can be used as a measure of similarity:

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

1 - Cosine_Similarity=Cosine_Distance

With 0 as very dissimilar and 1 very similar. -1 means they are facing opposite directions (180° angle).

2. Probabilities

Random Variable is a variable whose value depends on outcomes of random phenomena.

Probability Mass Function (PMF): assigns a probability to each value of a discrete random variable. The PMF describes a **Probability Distribution**.

Independent random variables can be **added** for **OR** and **multiplied** for **AND**.

Dependent variables usually offer a **joint probability P(Y, X)** and a **conditional probability P(Y|X)**, which is read “Prob of Y given X”.

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

$$P(X,Y) = P(Y,X) = P(Y|X)P(X)$$

3. Data Visualization

Helps with extracting information out of data. Extract possible relationships.

Seaborn and matplotlib are very powerful data visualization libraries for python.

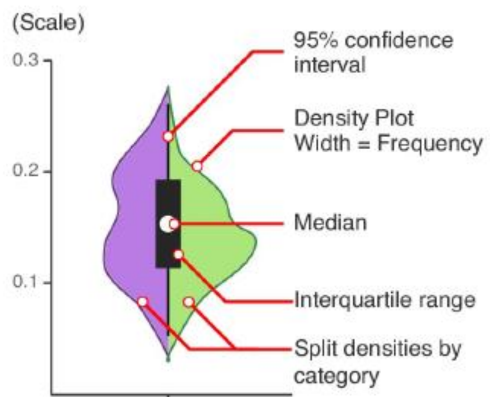
Line Plot: Nice for time series. Show change of one dimension relative to other. Examples: Prices, Population

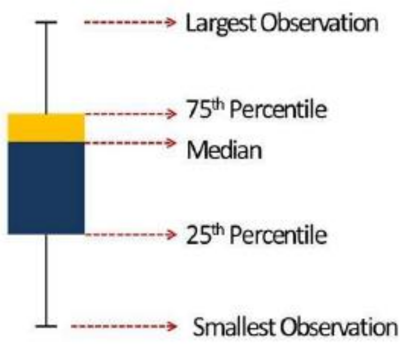
Bar Chart: Categorical data with bins. Examples: Apps per store

Histogram: Empirical distribution by automatically creating (equal width) bins. Example: Total bill categories plotted against the number of customers with that bill category.

Scatter Plot: Relationship between two continuous variables. Colour coding can add third dimension. Examples: Weight and height (with male and female as colour).

Box Plots and Violin Plots:





4. Linear Regression

Build a model that assumes a linear relationship between the predictors and the response.

$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \ldots + \beta_p x_{ip}$ Find best parameters B0 (intercept) and B1 to Bp (intercept) that minimize the loss (Mean Squared Error MSE):

$$e_i = y_i - \hat{y}_i \quad \text{Th}$$

$$E = \frac{1}{2N} \sum_{i=1}^{N} e_i^2$$

The residuals should be scattered around zero. If not, data is not linear

Pearson Correlation Coefficient: Tells you how strong the linear relationship is (-1 to 1). Will give zero for any other kind of relationship. Does not make any statement about causality.

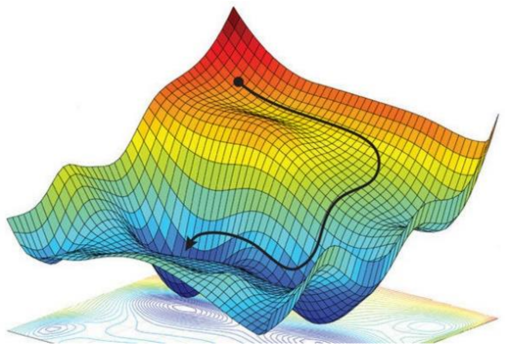
5. Optimization

Four key ingredients of Machine Learning:

Data; Cost Function (Loss); Model; Optimization

Optimization is the problem of finding a set of inputs to an objective function that results in minimum or maximum function evaluation.

Gradient Descent is an algorithm that finds an analytical solution to an optimization problem. It works by stepping towards valleys in the error surface and therefore finding local minima. This is achieved by computing the gradient of the error function at points and then stepping in the opposite direction (towards a valley). This is done until a criterion is met.



Visualization of the gradient stepping down the error surface

$$\theta^{new} = \theta^{old} - \underset{\boxed{\alpha = \textit{step size} \text{ or } \textit{learning rate}}}{\underset{\uparrow}{\alpha}} \nabla_\theta J(\theta)$$

Gradient Descent

Theta: set of params

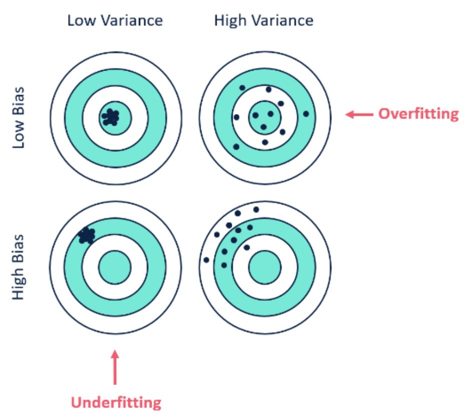
Stochastic Gradient Descent is a faster alternative that only uses a random subset of samples to calculate the gradient for each iteration. This leads to a fuzzier loss decrease but is significantly faster. For a fixed learning rate, SGD does not converge. For that reason, the learning rate is reduced over time, which is called **(simulated) annealing**.

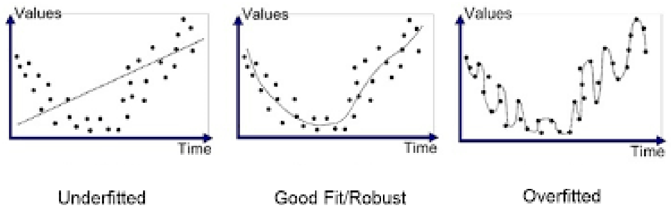
6. Regularization

Bias: The error that a model makes because of an erroneous assumption in the learning process.

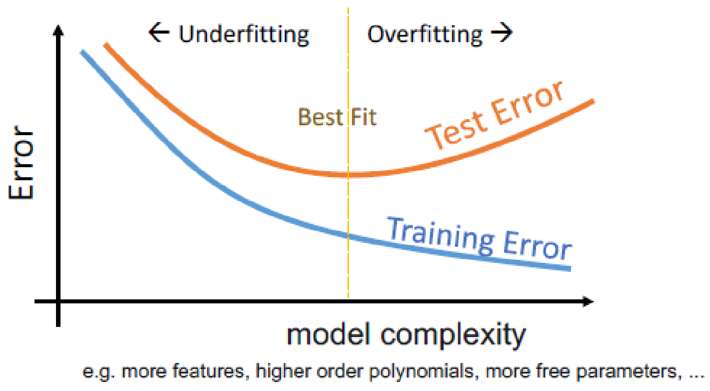
Variance: Variability of the model predictions. How strong the predictions change when some data of the same origin is removed or added.

Bias-Variance Trade-off: Changing one will always result in the other changing in the other direction. Fundamental aspect of designing a model.





A model's bias comes from the assumptions it makes about the true underlying nature of the data. A simple linear regression can never fit anything but a line to the data. This will lead to a high error (high bias) if the data is not actually linear in nature.



More complex model will always achieve better training results because they can use their flexibility to perfectly adapt to training data. This makes them lose their ability to generalize and thus increase the test error. The perfect model allows enough flexibility to capture the important patterns in the data but not enough to suffer from overfitting.

Regularization: Technique that discourages flexible and complex models to avoid overfitting. Overfitting happens by allowing the model to "memorize" the data exactly and with that adapt to the noise in it, which leads to high variance and low bias. Regularization significantly decreases variance without substantial increase in bias. This is achieved by introducing a term to the error calculation that discourages high coefficient values in regressions.

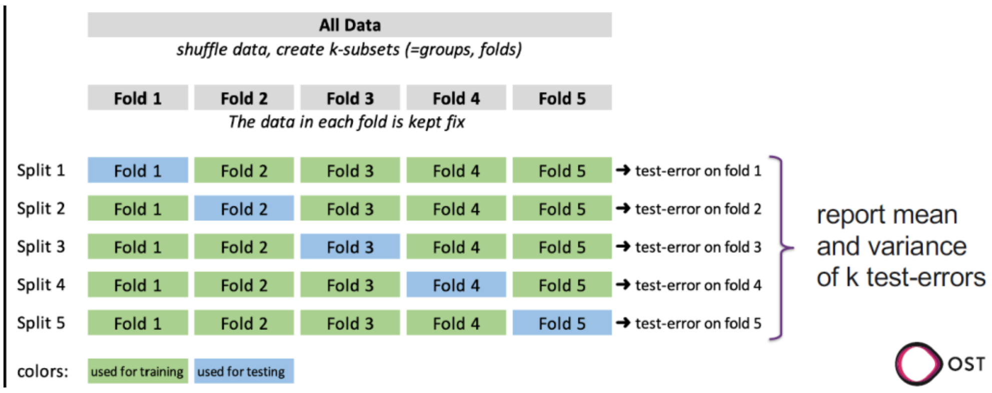
L1 Norm $\sum_{j=1}^{p} |\beta_j|$: Leads to more zero coefficients

L2 Norm: $\sum_{j=1}^{p} \beta_j^2$ Leads to more near zero coefficients

7. Cross Validation

k-fold Cross Validation is a technique with two broad use-cases:

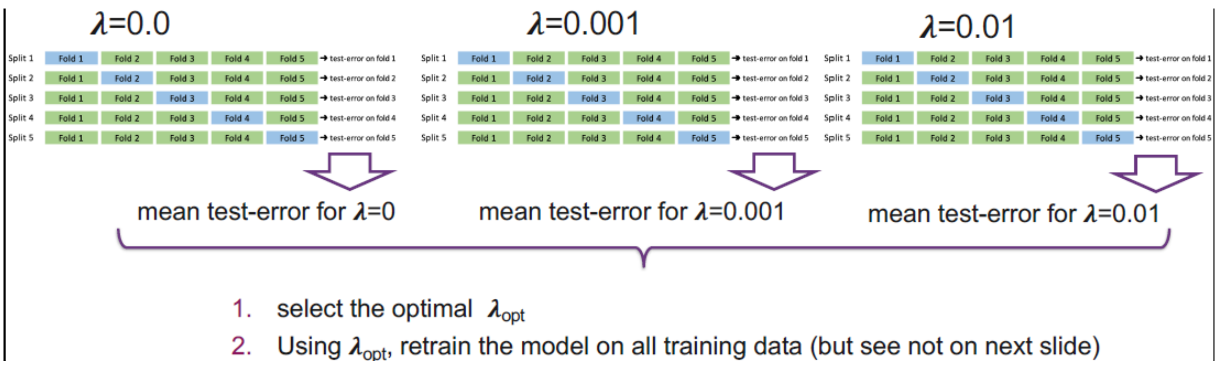
1. Get a good estimate of the generalization error



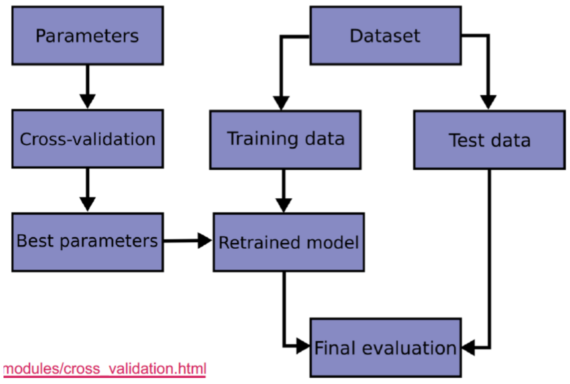
The dataset is split into k subsets. Each set is used once as test-set and k-1 times as training set. The model is trained and tested k times. The scores are aggregated to give a good estimate of the true generalization error.

2. Compare different model hyperparameters

Hyperparameters: Parameters of the training procedure that won’t appear in the model itself. For example, the learning rate of the gradient descent when training a regression model.


λ=0.0
λ=0.001
λ=0.01
mean test-error for λ=0
mean test-error for λ=0.001
mean test-error for λ=0.01
1. select the optimal λopt
2. Using λopt, retrain the model on all training data (but see not on next slide)

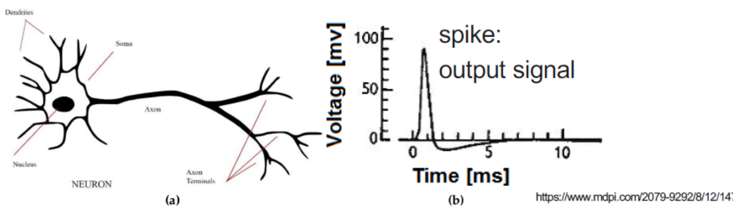
Different models with different hyperparameters are each tested using cross-validation. The best performing model is then trained again with the whole training set and validated with a separate subset the model has never seen before.



modules/cross_validation.html

8. Artificial Neural Networks (ANN)

ANNs consist of layer of artificial neurons that are inspired by the real neurons in our brains.

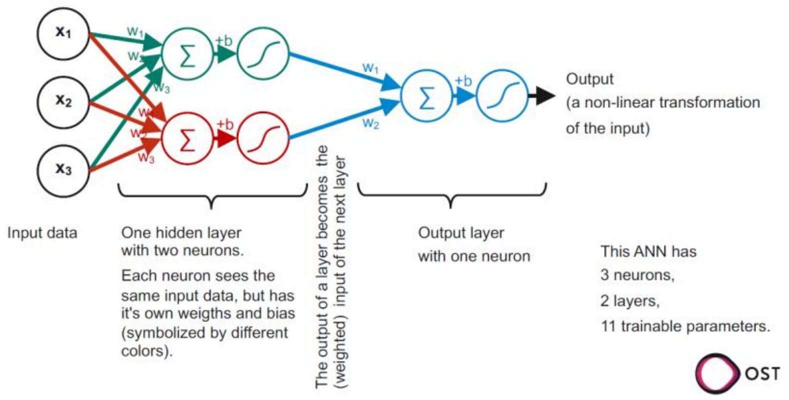


- *Dendrites*: Collect (chemical) **input** from (many) other neurons.
- *Cell Body (Soma)*: If the collective input signal reaches a (voltage) threshold, an **output** signal, called a "spike" is generated.
- *Axon*: the electrical pulse (spike) travels along the axon to other neurons.

Neurons receive input from other neurons and fire if a certain input threshold is met. Learning means strengthening existing and building new connections (synapses) between neurons.

Although the individual neurons do very simple computations, a connected system of many neurons can perform highly complex computations.

ANNs are inspired by this:


x1
x2
x3
w1
w2
w3
+b
Σ
w1
w2
Output
(a non-linear transformation
of the input)
Input data
One hidden layer
with two neurons.
Each neuron sees the same input data, but has it's own weigths and bias (symbolized by different colors).
The output of a layer becomes the (weighted) input of the next layer
Output layer
with one neuron
This ANN has
3 neurons,
2 layers,
11 trainable parameters.
OST

Training the ANN means finding the right weights for every neuron. This is done via an algorithm called **Backpropagation** that calculates the partial derivative for every single weight very efficiently.

ANNs with multiple hidden layers (layers between input and output) are called deep neural networks.

Early layers of ANNs can identify big patterns (such as light or dark areas). Later layers use this to identify mor complex and fine-grained patterns (such as lines and shapes). The output layer combines all findings to an output.

9. Binary Classification with Logistic Regression

Assigning a binary value to unknown observations consisting of features. This is done by defining a value that outputs a value between 0 and 1 that can be interpreted as the likelihood of this observation belonging to class 1.

$$Pr(y = 1|\mathrm{x}; W) = \Pr(y = 1 \mid \mathrm{x}) = p(\mathrm{x}) = \frac{1}{1 + e^{-(\mathbf{W}^T\mathbf{x})}}$$

For this the **sigmoid function** is used. A weighted sum of features is then put in and the function is evaluated. A probability exceeding a certain threshold is then classified as class 1, else class 2.

$$Minimize\ cost(W) = \frac{-1}{N}\sum_{i=1}^{N}(\boldsymbol{y_i} * \log(p_i)) + (\boldsymbol{1} - \boldsymbol{y_i}) * \log(1 - p_i))$$

By maximizing this equation, we find the weights that are most likely to have produced the training set. Because this equation is convex, we can use gradient descent to find its minimum.

10. Classifier with k nearest neighbours (KNN)

KNN assigns the labels of the majority of the k nearest neighbours to the new observation. KNN can deal with not linearly separable classes. It forms arbitrary complex **decision boundaries** (values for which the probability equals 0.5).

It does **not require training** and is a **non-parametric** approach. Can also deal with **multiple classes**.

It is **not well suited for high dimensional data** because the distances become very large. Data should be properly **scaled** before.

KNN Algorithm:

Step-1: Select the number K of the neighbours

Step-2: Calculate the distance of K number of neighbours

Step-3: Take the K nearest neighbours as per the calculated distance.

Step-4: Among these k neighbours, count the number of the data points in each category.

Step-5: Assign the new data points to that category for which the number of the neighbour is maximum.

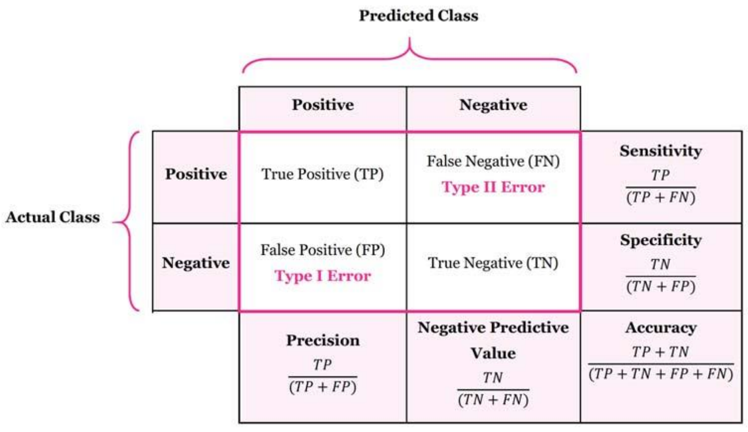
The hyperparameter K influences bias and variance. The higher K, the lower the variance but the higher bias. Cross Validation helps with choosing the best.

Cosine distance, $cost\ \theta = \frac{\mathbf{x_1} \cdot \mathbf{x_2}}{||\mathbf{x_1}||\ \ ||\mathbf{x_2}||}$

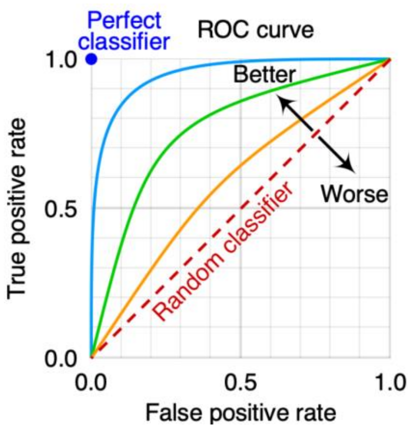
Manhattan Distance , $d_M = \sum_{i=1}^{n} |x_{1,n} - x_{2,n}|$

Euclidean Distance, $d_E = \sqrt{\sum_{i=1n}(x_{1,n} - x_{2,n})^2}$ Distances can be either of these

11. Classifier Evaluation



		Predicted Class		
		Positive	Negative	
Actual Class	Positive	True Positive (TP)	False Negative (FN) Type II Error	Sensitivity $\frac{TP}{(TP + FN)}$
	Negative	False Positive (FP) Type I Error	True Negative (TN)	Specificity $\frac{TN}{(TN + FP)}$
		Precision $\frac{TP}{(TP + FP)}$	Negative Predictive Value $\frac{TN}{(TN + FN)}$	Accuracy $\frac{TP + TN}{(TP + TN + FP + FN)}$



Confusion Matrix; Sensitivity = Recall = TPR; FPR = FP / (FP + TN); Miss Rate = FNR = 1 - TPR

Threshold influences these values, is management decision. Higher Threshold -> more false Negatives; lower -> more false Positives

ROC Curve: Area under curve should be as close to 1 as possible for classifier to be good.

12. Clustering (K Means)

Unsupervised Algorithm that aims to identify clusters of similar observations that are similar to each other but dissimilar to other clusters.

K-Means Algorithm:

1. Let us assume we know the number of clusters k_c
2. Initialize the value of k cluster centres (aka, means, centroids) $(\mathbf{C_1}, \mathbf{C_2}, \ldots\ldots \mathbf{C}_{k_c})$
3. Assignment :
   1. Find the **squared Euclidean distance** between the centres and **all the data points.**
   2. Assign each data point to the cluster of the **nearest centre.**
4. Update: Each cluster now potentially has a new centre (mean). Update the centre for each cluster
   1. New Centres $((\mathbf{C'_1}, \mathbf{C'_2}, \ldots\ldots \mathbf{C'}_{k_c})$ = Average of all the data points in the cluster$(1,2,\ldots,k_c)$
5. If some stopping criterion met, Done
6. Else, go to Assignment step 3

Stopping criteria: assignments do not change; Distance to centres is below threshold; max number of iterations done

Evaluation of KNN models is hard because they are unsupervised. Two scores can be computed:

Inertia or **within-cluster sum-of-squares (WCSS)**; lower is better: $\sum_{k=1}^{k_c} \sum_{\mathbf{x_i} \in Member(C_k)} d(C_k, \mathbf{x})$

Silhouette Score: $\frac{b-a}{\max(a,b)}$ Range from -1 to,1; should be 1

With a: average intra-cluster dist. (avg dist. Between each point); b: average inter-cluster distance (distance btw. Cluster and nearest neighbour cluster)

13. Ensemble Methods

Ensemble Methods combine multiple different models to one whose performance is better than the best individual one. It utilizes the **wisdom of crowd**. **No free lunch** theorem says no individual model is universally the best-performing for all problems. Hence using many different models helps.

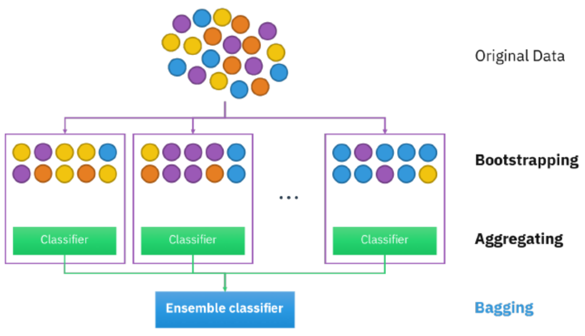
Models can differ in their Algorithms (e.g., KNN vs. Log. Regression), the hyperparameters or the training data. **Random Patches**: Varying features and data; **Random Subspaces**: only varying features.

Prediction Learners can be aggregated with

1. Hard Voting: Predict the class that gets the most votes

2. Predict the class the with the highest average class probability

Bagging: Sample observations **with** replacement; **out of bag (oob)** data: data that has not been selected when bagging can be used as validation set



Pasting: Sample observations **w/o** replacement

Pasting and Bagging can be done with features and data.

Bootstrapping: Make multiple datasets out of one