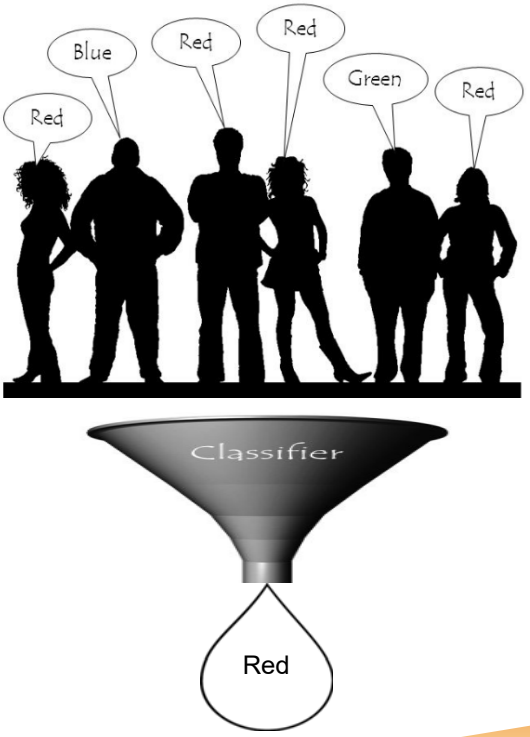
Ensemble Methods



Dr. Barry Shepherd
Institute of Systems Science
National University of Singapore
Email: barryshepherd@nus.edu.sg

What are Ensembles?

- Ensembles are committees of multiple models
- Each model makes a prediction or "vote"
- Final prediction is average/majority of votes
  - Majority vote for classification (class prediction)
  - Average prediction for regression problems (value prediction)

- What benefits does this bring?
- Why not just train one smart model?

Motivation

- Assume one model and 5 test cases

Truth	1	0	1	1	0	Accuracy
Model 1	1	0	0	1	1	60%

Motivation

- Add another 5 models, each with same accuracy, but with variance (models do not give identical predictions)

Truth	1	0	1	1	0	Accuracy
Model 1	1	0	0	1	1	60%
Model 2	0	1	1	1	0	60%
Model 3	0	0	1	0	0	60%
Model 4	1	1	1	1	1	60%
Model 5	1	0	0	0	0	60%
Vote 1-5	1	0	1	1	0	100%

- No one model is very accurate, learns everything
- Performance of ensemble outperforms individuals
- Usually more reliable/robust than individual models

What makes a good ensemble?

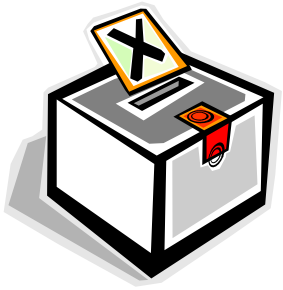
- The term *ensemble* is usually reserved for methods that generate multiple hypotheses using the **same** base learner*

> *"A necessary and sufficient condition for an ensemble of classifiers to be more accurate than any of its individual members is if the classifiers are accurate and diverse"*
>
> -- Tom Dietterich (2000)

(*the broader term of ***multiple classifier systems*** covers ensembles that do not use the same base learner)

How to get suitable diverse models?

- *Ensembles* tend to yield better results when there is a significant diversity among the models
- Bagging is one way of introducing diversity
  - Train many models with different random samples
  - Usually applied to decision trees, but can be used with any method

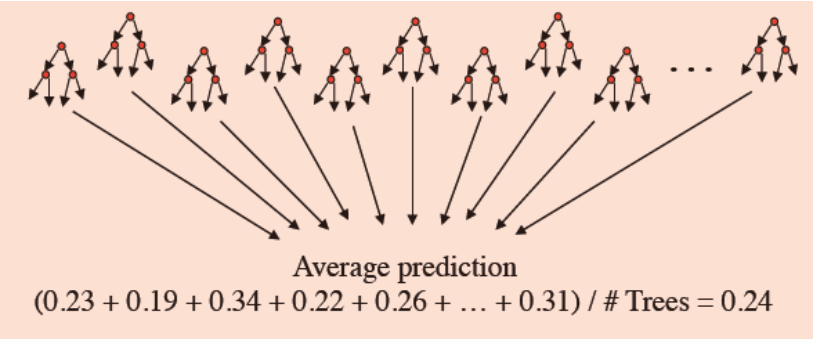


When using bagging, each learned model has a vote of equal value

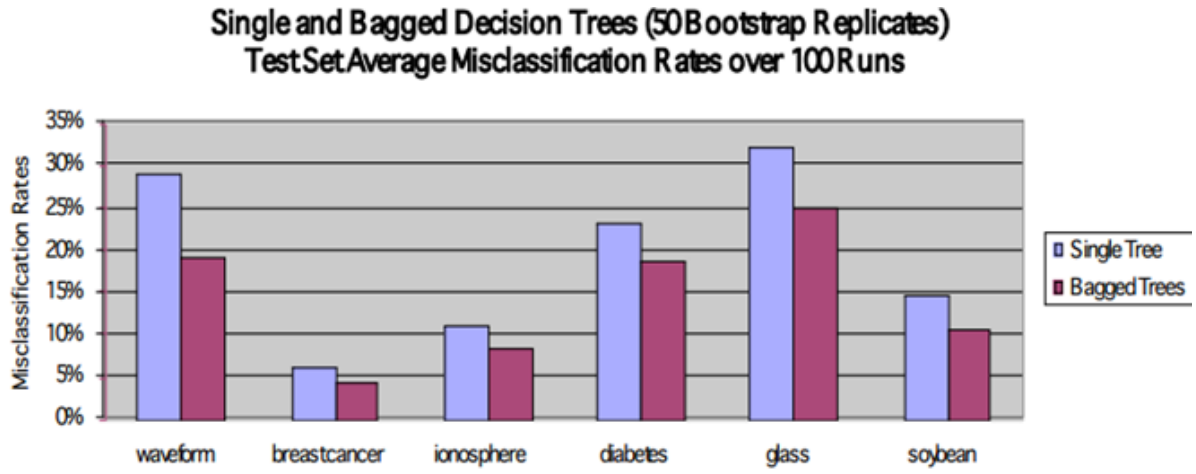
- Where do we get many different random samples?

Bagging: Bootstrap Aggregating

- Draw N (say 100) bootstrap samples (sampling with replacement) from the training data, Train decision trees on each sample
- Algorithm:
  - Randomly draw 67% (say, two thirds) of the training data
  - Train a tree on this sample
  - Repeat this N times to get N trees
- Take the un-weighted average prediction of all trees

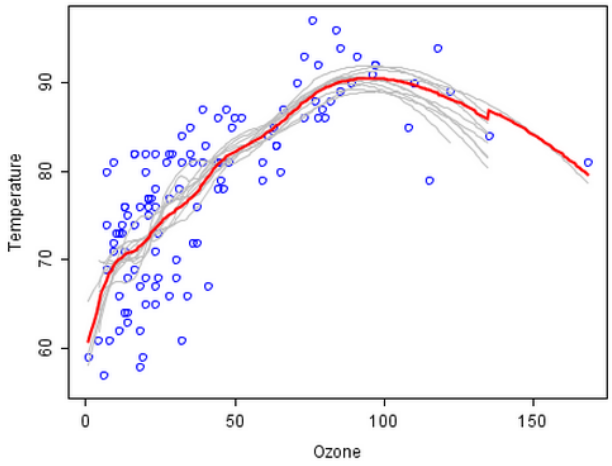


Examples of Bagging (Breiman, 1996)



Examples of Bagging

- Ensemble of 100 LOESS* predictors built from 100 bootstrap samples

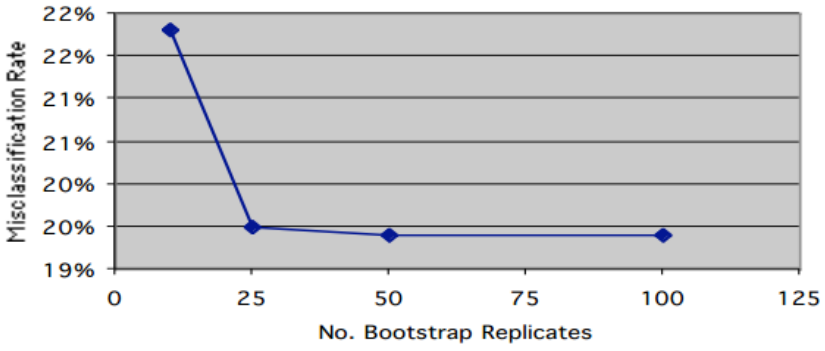


(data from Rousseeuw and Leroy (1986)).

*LOESS (local regression) ~ build a mini regression model for each training point using weighted least squares, giving more weight to points near the training point. Smoothly join the individual mini-models to get a single classifier (an example of a "kernal" method)

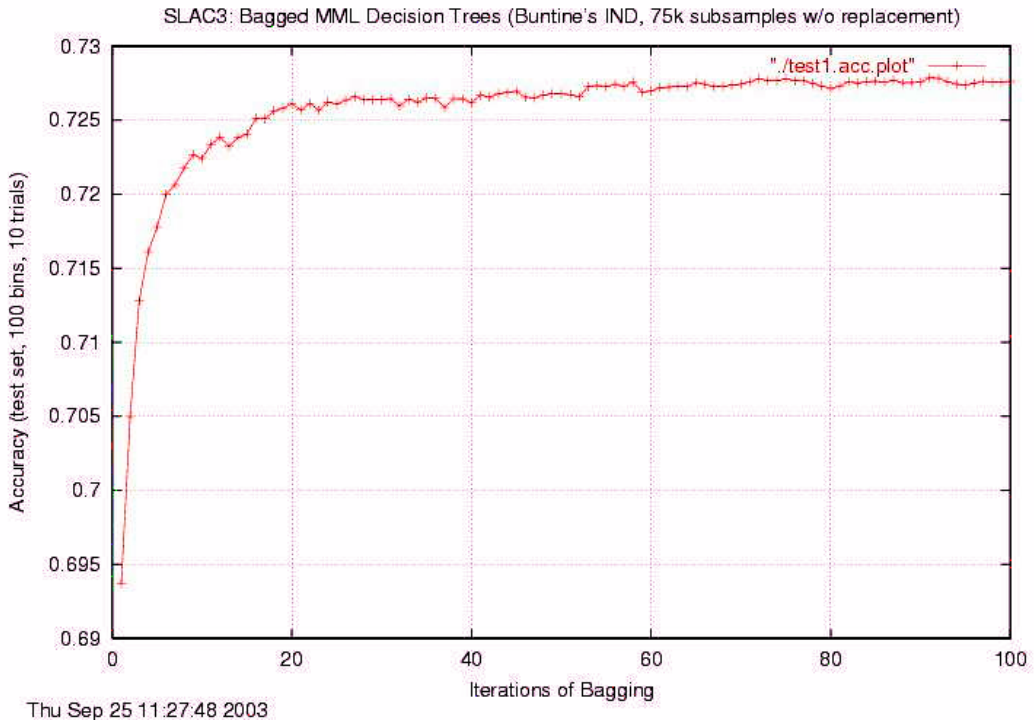
How Many Bagged Models Are Required?

- Example from the soybean data set , Breiman, 1996:



- Depends on data and problem, but generally, < 50 models should work well and often < 25 is adequate

How Many Bagged Models Are Required?

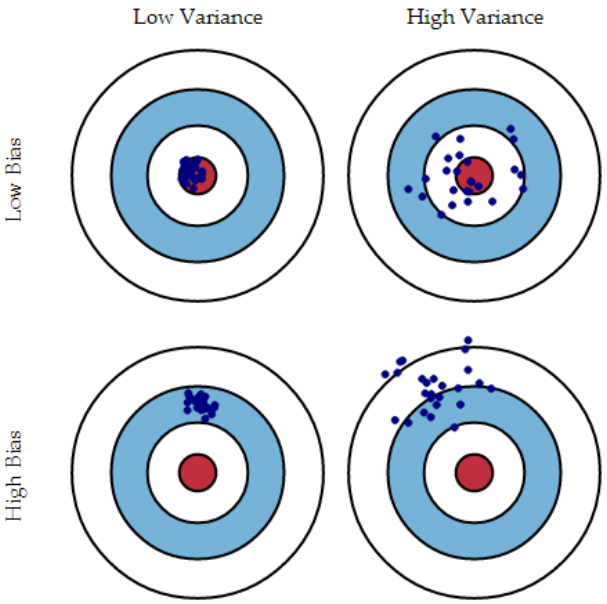


Why does averaging a set of DTs help?

- Wouldn't it be better if we
  - Use all of the training data to build a single good tree?
  - Improve the learning algorithm to build a better single tree?
- Not really…
  - Decision tree cuts are harsh
  - Do not produce a very smooth decision surface
  - Even slightly different training samples can give you a different tree
- Ensembles + DTs work well together…

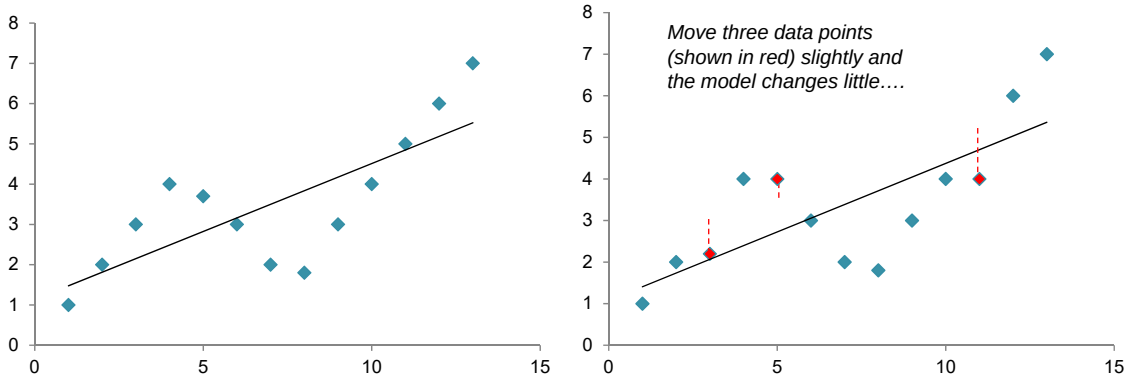
Why does it work? Types of Model Error

- **Assume**
  - You sample a population N times and build a model from each sample, then…
- **Error due to Bias:**
  - The difference between the expected (or average) prediction of the model and the correct value
- **Error due to Variance:**
  - The variability of the model prediction for a given data point



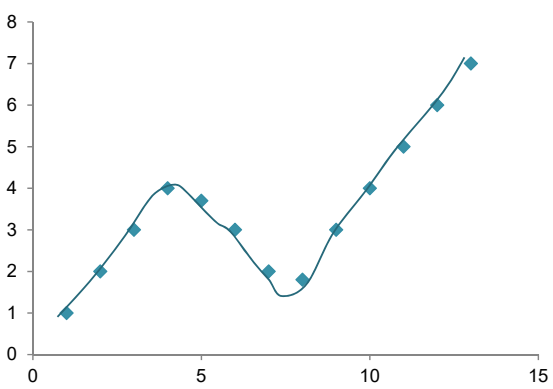
Types of Model Error

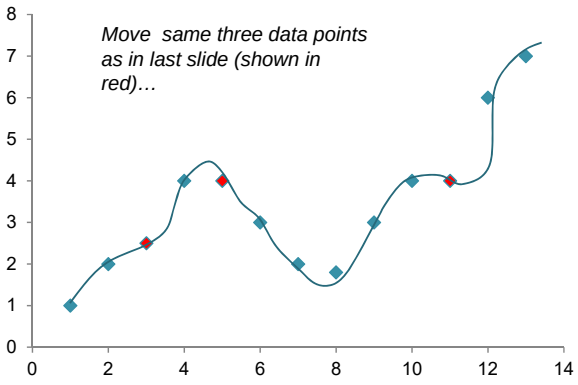
- Models that under-fit the data tend to have:
  - High bias ~ the model doesn't fit the training data very well
  - Low variance – removing/changing a few training data points won't change the model or predictions much



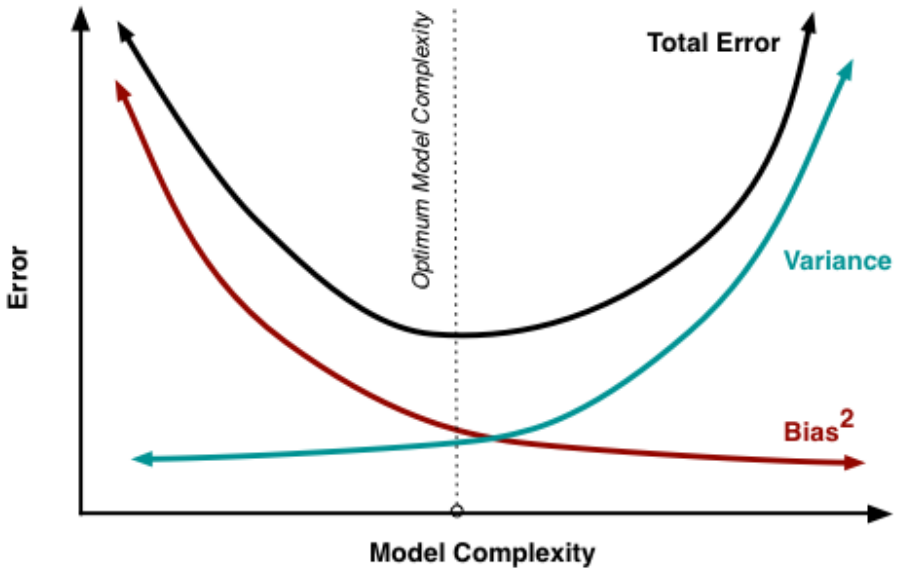
Types of Model Error

- Models that over-fit the data tend to have:
  - Low bias ~ the model fits the training data very well
  - High variance – removing/changing a few training data points can change the model and hence the predictions a lot





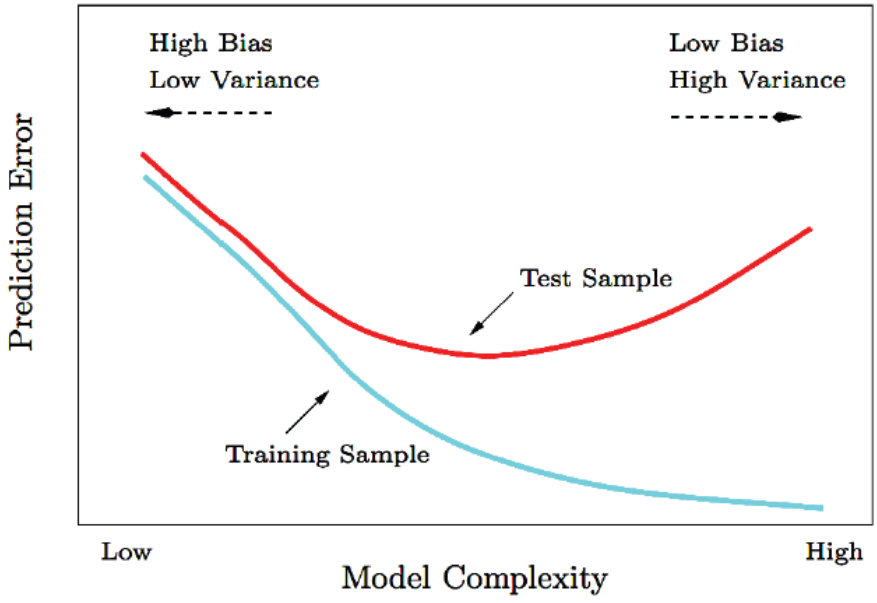
Tradeoff between Bias and Variance



Tendency to overfit

Tradeoff between Bias and Variance

- Over-fitting cannot be detected from the model performance on the training data



Hastie, Tibshirani, Friedman "Elements of Statistical Learning" 2001

Tradeoff between Bias and Variance

- Reducing bias & variance is important for prediction accuracy
- Tradeoff:
  - bias vs. variance
  - high complexity models vs. low complexity models
  - most errors due to over-fitting vs. most errors due to under-fitting
  - choice: smart twitchy (sensitive) models vs. less smart but stable models
- Clearly we want smart models, but…
  - Can we reduce variance without increasing bias?
  - Can we reduce over-fitting without under-fitting?

YES!

Reduce Variance Without Increasing Bias

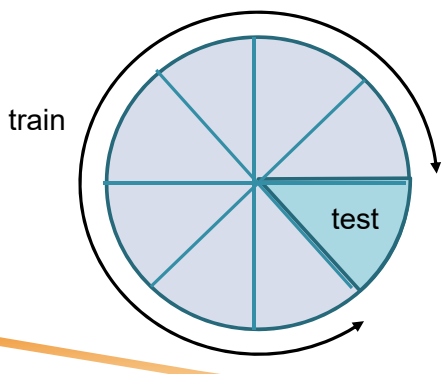
- Averaging reduces variance:

$$Var(\overline{X}) = \frac{Var(X)}{N}$$

- Average across models to reduce model variance
- For large N, residual model error mainly due to bias!
- In practice:
  - models are correlated, so reduction is smaller than 1/N
  - variance of models trained on fewer training cases usually larger
  - only works with some learning methods: very stable learning methods have such low variance to begin with, that bagging does not help much

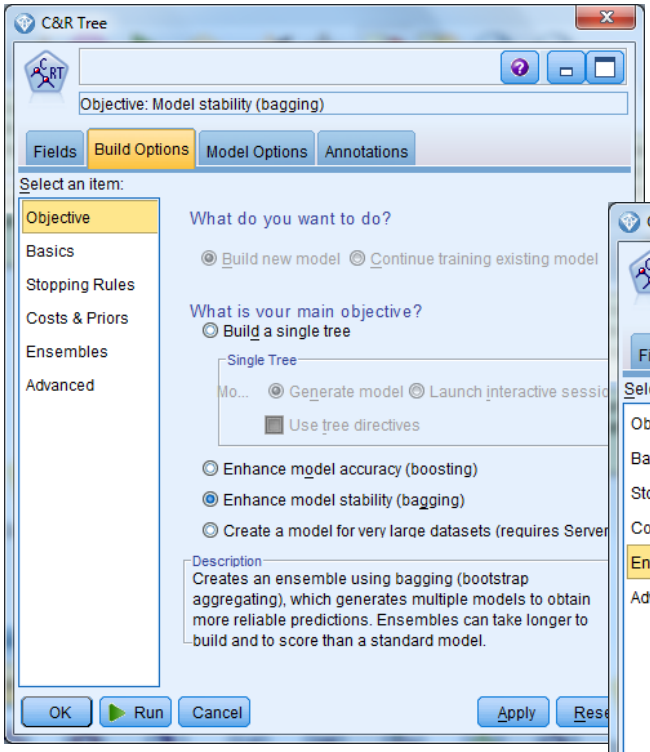
Bagging using Cross-Validation

- Train N models (e.g. Neural Nets) using N-fold cross-validation
  - Train on N-1 folds and stop training early
  - Use held-out fold to do parameter optimization
  - Repeat for all sets of N-1, At the end you have N models
- Then, instead of retraining on all data with new parameters, keep N models and average predictions
  - Often safer/easier than retraining from scratch

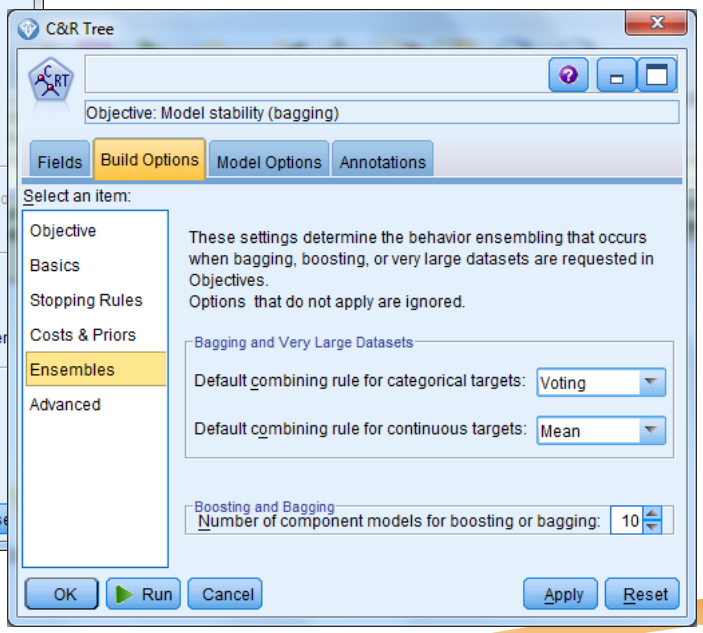


e.g. 8-way
cross-validation

Bagging in SPSS Modeler



Some model nodes implement bagging & boosting



Other ways to create Model Variance

- Manipulating the training data (e.g. bagging)
- Manipulating the input features
- Varying the classifier type, architecture

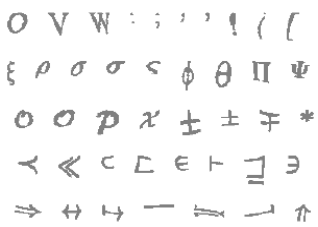
Random Forests

- Draw *1000+* bootstrap samples of data
- ***Draw random sample of available features at each tree split***
  - Randomisation (hence model diversity) now occurs in two places:
    Random sampling of *training data* + Random sampling of *feature set*
  - Training speed increases due to less computation at each tree split (less features to evaluate the splitting cost function for)
- Train trees on each sample/attribute set -> *1000+* trees
- Use un-weighted voting to get final prediction (as with bagging)

Random Forests

- Usually works better than bagging
  - robust to noise, easy to use, surprisingly high accuracy
  - but.. lots of trees means hard to interpret (becomes a black box)
- Variance of RF trees is higher than Bagged Trees
  - typically needs 10X as many trees
  - trees should be (generally) unpruned (to encourage diversity)
  - RF needs 100's to 1000's
- Extra parameter to tune: p(feat)
  - probability of getting to use feature at each split
  - fortunately, usually not too sensitive
  - Breiman suggests SQRT(N)
- Unlike Bagging and Boosting, RF is for trees only

Random Forests in Vision



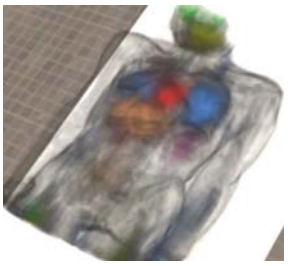
[Amit & Geman, 97]
digit recognition

[Lepetit *et al.*, 06]
keypoint recognition

[Moosmann *et al.*, 06]
visual word clustering

[Shotton *et al.*, 08]
object segmentation

[Rogez *et al.*, 08]
pose estimation

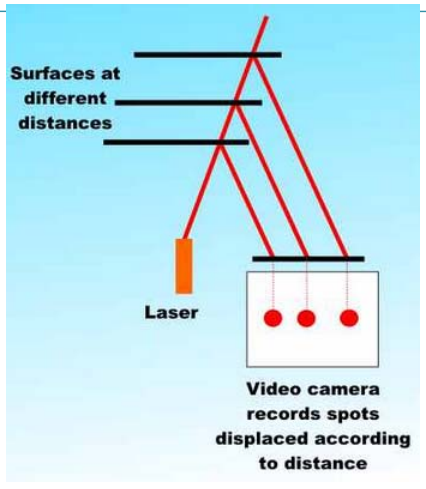


[Criminisi *et al.*, 09]
organ detection

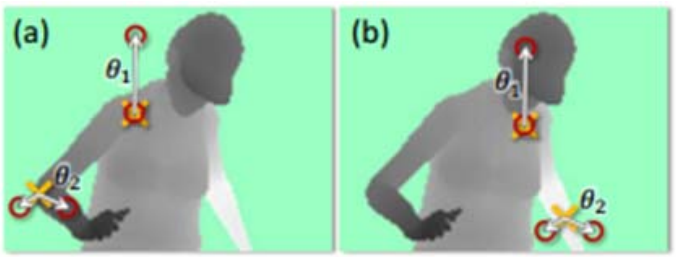
Among many others…. Microsoft Kinect Uses Random Forests

Kinect's Decision Forest

- Step1: Generate a 3D image
  - Kinect uses "structured light" ~ If you have a light source offset from a detector by a small distance then the projected spot of light is shifted according to the distance it is reflected back from.

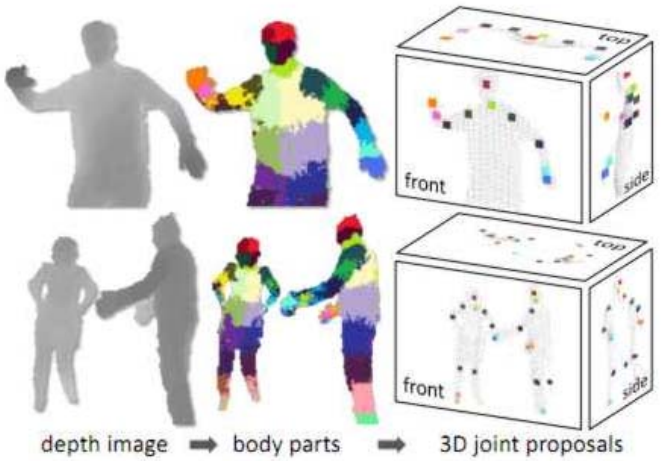


- Step2: Compute Features
  - Compute the difference in depth (z) to two pixels that are close together in (x,y). If difference is small then they likely belong to same object. Repeat many times.



Kinect's Decision Forest

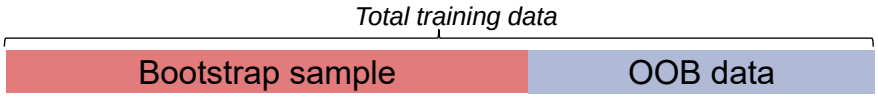
- Step3: Build Forest
  - Each tree was trained on features that were pre-labeled with the target body parts.
  - Training just 3 trees using 1 million test images took a day using a 1000 core cluster.
  - The trained classifiers assign a probably of a pixel being in each body part
- Step4: Execute the Forest
  - Picks areas of maximum probability for each body part type.



http://www.youtube.com/watch?v=HNkbG3KsY84

Testing Random Forests

- No need for separate test set
- Method:
  - Test each tree against the data left over after the bootstrap sample was taken; this is called the OOB (out-of-bag) data

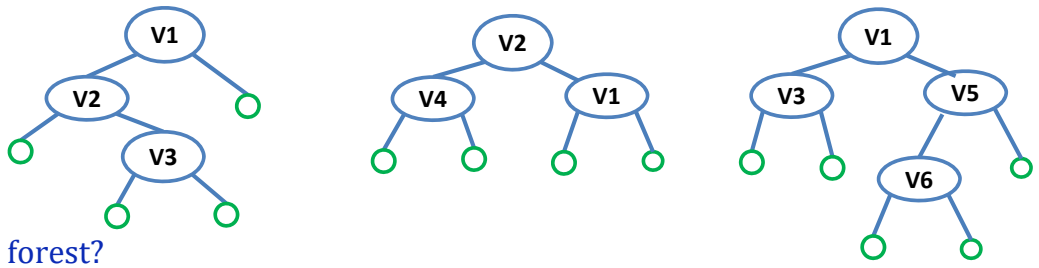


  - If each bootstrap sample takes 67% of the training data, then after building T trees every training example will have been OOB (and hence a valid test example) for about T/3 times.
  - For each training example, take the majority vote of all T/3 test predictions to get the forest's prediction* and compare with the actual class value. Output 1 if forest prediction != actual, else output 0
  - Sum over all training examples to get error estimate for the forest

** For regression problems, average all N/3 test predictions to get the forest prediction. Then compute MSE as $\Sigma_{training\ examples}$ (prediction – actual)**2*

Measuring Variable Importance

- For a single tree the order in which the variables occur in the tree is a measure of their relative importance to the prediction



- For a forest?
  - Naïve method: Count the number of times the variable occurs in all of the trees, more occurrences => more important
  - Better method: sum the total reduction in impurity (the decreases in the Gini index) for all nodes that test the variable

Measuring Variable Importance

- Permutation Method
  - Randomly shuffle the values of a given input variable to "break" the bond of the variable to the response. Then the difference of the model accuracy before and after the shuffling is a measure of how important the variable is for predicting the response
- Detailed Method
  1. Test every tree on its own OOB examples. For each training example e count the votes for the correct class (call this $\text{NormalCorrectVotes}_e$)
  2. For each input variable v:
     - For each tree t:
       - Randomly permute the values for v in the OOB examples and retest the tree
     - For each training example, count the votes for the correct class
     - $\text{Importance}_v = \text{average}\left[\frac{\text{NormalCorrectVotes}_e - \text{ShuffledCorrectVotes}_e}{\text{TotalVotes}_e}\right]$

Boosting

- Can a set of **weak learners** create a single **strong learner**?
  - A theoretical question that triggered much research in 1980's & 1990's
  - A weakly learned model is only slightly better than random guessing
  - A strongly learned model is arbitrarily well-correlated with the truth
- Boosting essentials:
  - Build a model (but don't 100% over-fit the data!)
  - Increase weights of the training examples the model gets wrong
  - Retrain the model using the weighted training set
  - Repeat many times…

Incorrectly classified examples count for more when the model is retrained

Basic Boosting Algorithm

1. Weight all training samples equally
2. Train model on train set
3. Compute error of model on train set
4. *Increase weights on train cases that the model gets wrong!*
5. Train new model on re-weighted train set
6. Re-compute errors on weighted train set
7. Increase weights more on cases it still gets wrong
8. Repeat until tired (100+ iterations)
9. Final model: *weighted* prediction of each individual model (aka base-models)

Most well-known & successful boosting algorithm is AdaBoost (Adaptive Boosting, *Freund and Schapire, 94*)

Recent popular algorithms: SMOTEboost, Gradient Boosting

AdaBoost* (Adaptive Boosting)

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in \mathcal{X}$, $y_i \in \{-1, +1\}$. ← The training examples

Initialize: $D_1(i) = 1/m$ for $i = 1, \ldots, m$. ← The training example weights

For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t. ← Train a model (build a classifier)
- Get weak hypothesis $h_t : \mathcal{X} \to \{-1, +1\}$.
- Aim: select h_t with low weighted error:

$$\varepsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i].$$

← Goal of classifier is to reduce weighted error relative to Dt

← Cases where the prediction does not equal the real class value

- Choose $\alpha_t = \frac{1}{2} \ln\left(\frac{1 - \varepsilon_t}{\varepsilon_t}\right)$.
- Update, for $i = 1, \ldots, m$:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

← Re-weight the examples

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

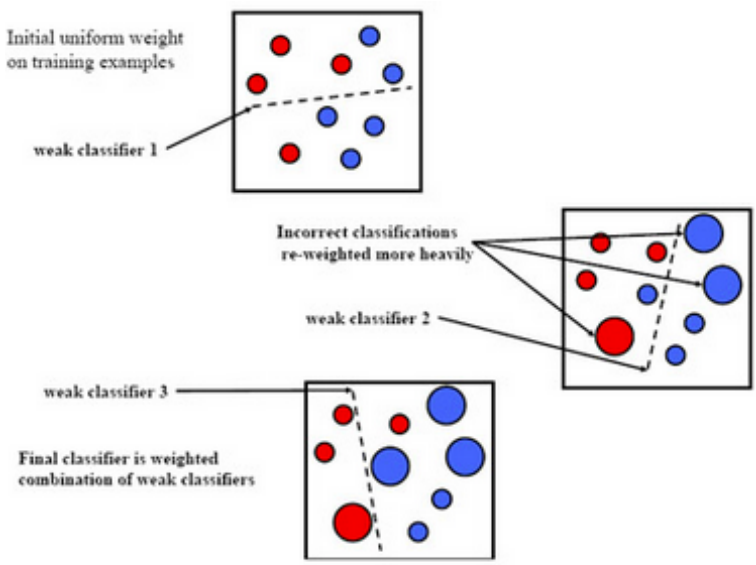
Output the final hypothesis:

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right).$$

← The "final" prediction is the weighted average of all of the weak classifiers

Freund and Schapire, 94

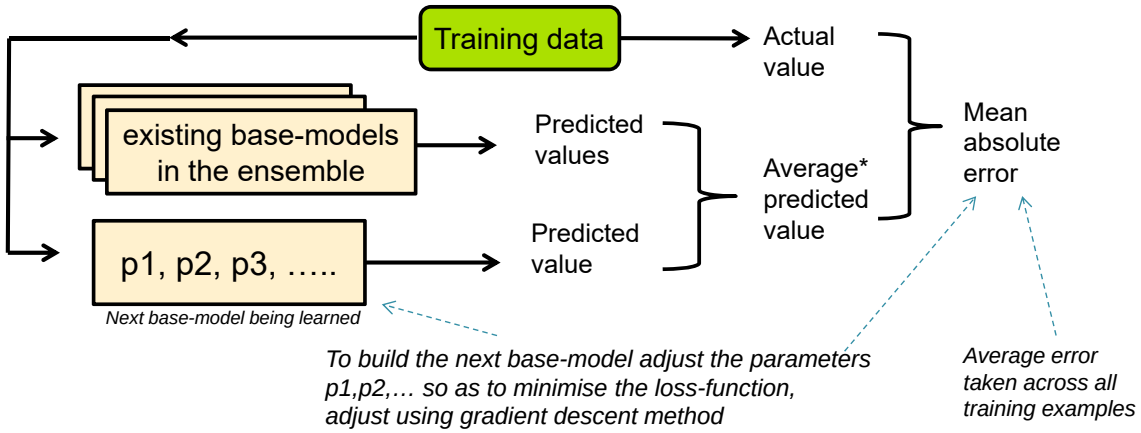
AdaBoost - Conceptual



$$H(x) = sign(\alpha_1 h_1(x) + \alpha_2 h_2(x) + \alpha_3 h_3(x))$$

Gradient Boosting Machine (GBM)

- Treat learning as Gradient Descent optimisation
- Express the model (i.e. each base-learner) as a set of parameters to optimise subject to an objective function, called a loss-function, e.g. MAE (mean abs. error)
  - E.g. For NN, the parameters are the weights and the number of nodes and layers
  - E.g. For decision tree, the parameters could be a set of (attribute index, threshold) values
- Like Adaboost, models are learned and added to the ensemble sequentially



XGBoost is a popular Gradient Boosting Library

A good background read!

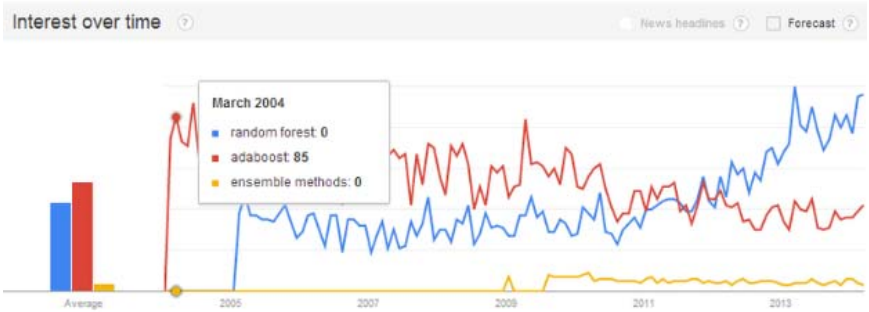
March 25, 2014

A Thumbnail History of Ensemble Methods

By Mike Bowles

Ensemble methods are the backbone of machine learning techniques. However, it can be a daunting subject for someone approaching it for the first time, so we asked Mike Bowles, machine learning expert and serial entrepreneur to provide some context.

Ensemble Methods are among the most powerful and easiest to use of predictive analytics algorithms and R programming language has an outstanding collection that includes the best performers – Random Forest, Gradient Boosting and Bagging as well as big data versions that are available through Revolution Analytics.



See http://blog.revolutionanalytics.com/2014/03/a-thumbnail-history-of-ensemble-methods.html

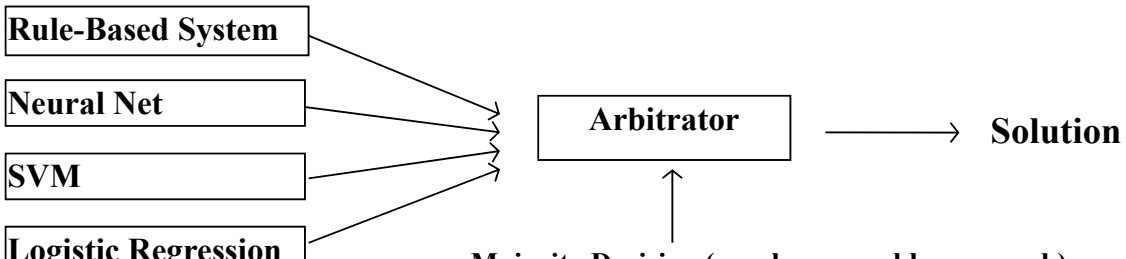
Multiple Classifier Systems (MCS)

- In an Ensemble, the models cooperate to make a prediction
  - Ensembles work best when all models are of the same type
  - Having lots of ensemble members works best

- In "Mixture of Experts" approach, each classifier is expert in certain situations (Jordon, Jacobs, 1994)
  - Each model type has different strengths and weaknesses
  - Usually have relatively small number of experts

Mixture of Experts Example

- Different solution strategies (experts) offer alternative solutions. Another process decides which solution to accept or how to combine the solutions, e.g. majority vote algorithm. This architecture is also known as stacking*



Individual Experts

Majority Decision (regular ensemble approach)
If majority agree then good decision, else unreliable decision

OR…

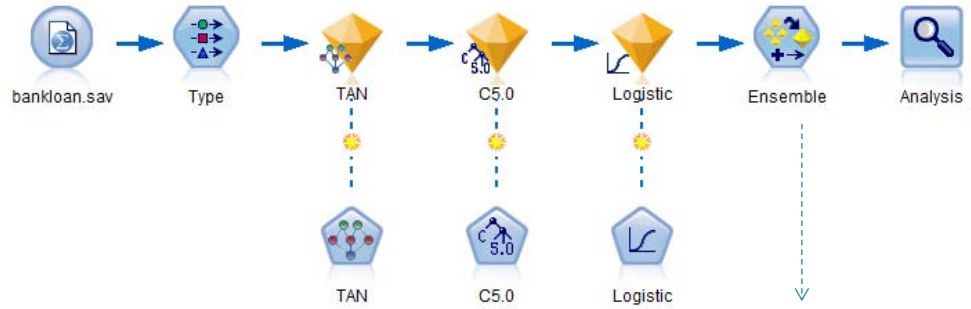
Weighted Decision
Weight expert judgement's according to circumstances

Best Expert Only
Decide which expert is most appropriate for current situation

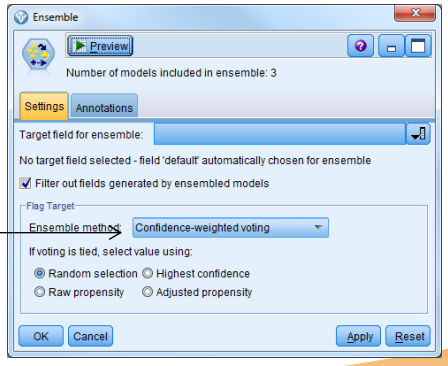
**Stacking (sometimes called stacked generalization) involves training a learning algorithm to combine the predictions of several other learning algorithms (Wikipedia)*

Multi-Classifier Systems in SPSS

- The "Ensemble" node allows MCS to be built.
  (Bagging and boosting is handled within the modelling nodes)

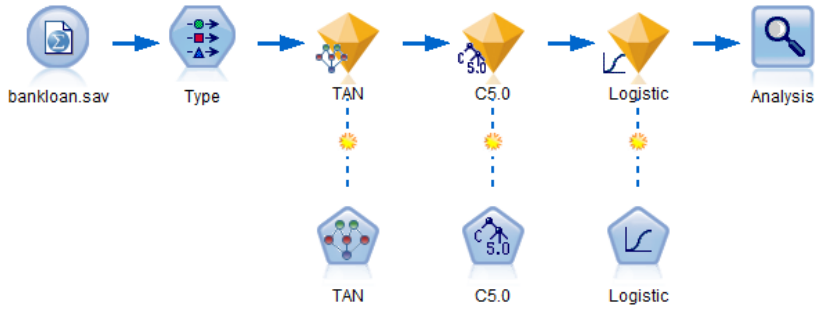


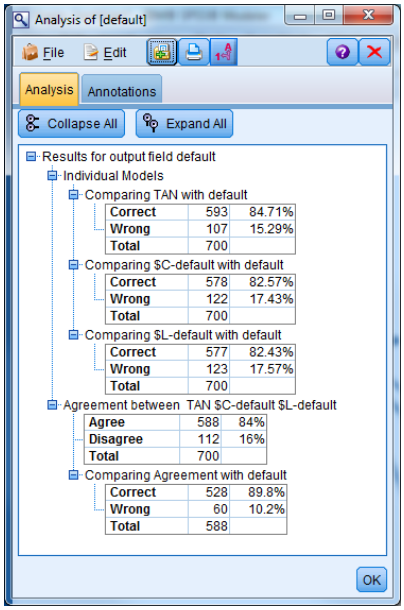
Ensemble method	Field name
Voting Confidence-weighted voting Raw-propensity-weighted voting Raw-propensity-weighted voting Highest confidence wins	*$XFC_<field>*
Average raw propensity	*$XFRP_<field>*
Average adjusted raw propensity	*$XFAP_<field>*



Multi-Classifier Systems in SPSS

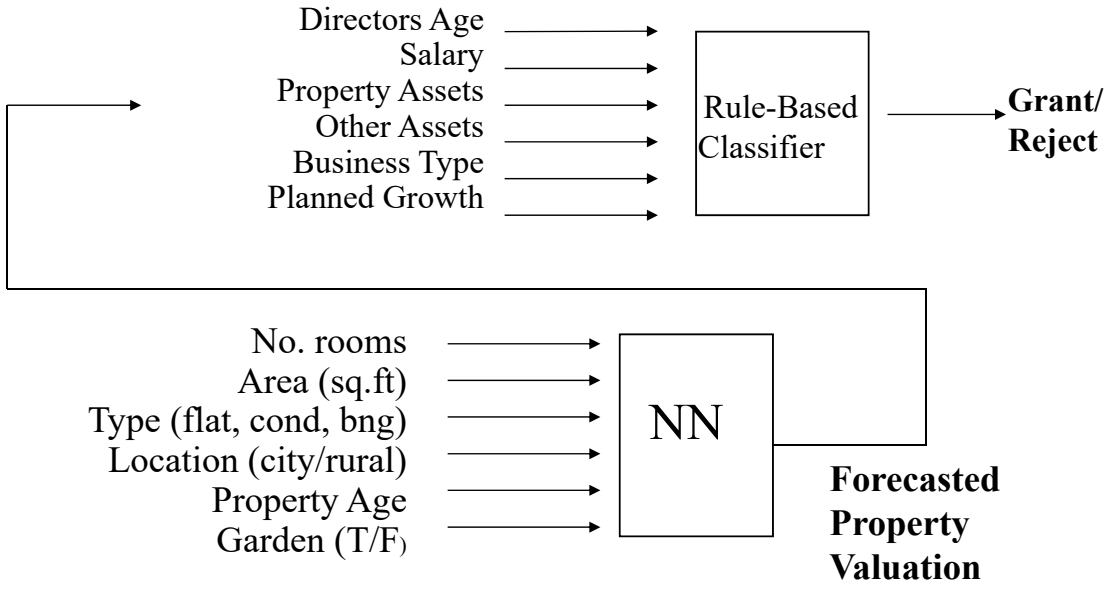
- Chaining together models without an Ensemble node only allows the agreement between models to be seen – no voting is performed!





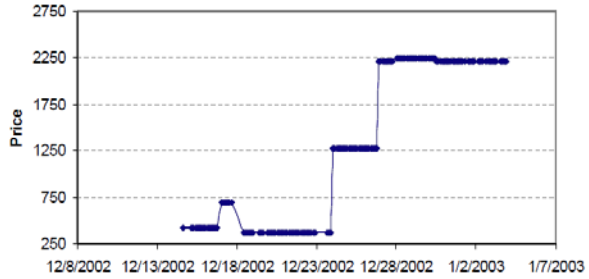
Mixture of Experts Example

- Approving business loan to a small company
- In this example the Experts have different skills

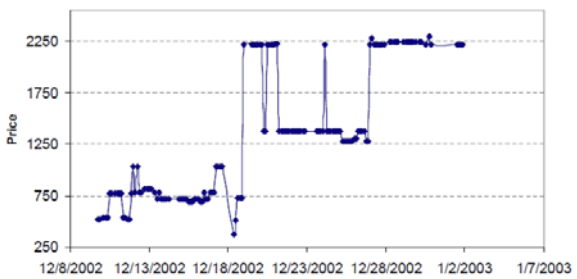


Mixture of Experts Example: Airfare Prediction

- Airlines operate proprietary pricing rules that are unknown to consumers & use hidden variables (e.g. number of unsold seats)
  - US flight prices can fluctuate up & down during the day and increase as the flight date gets nearer. Changes depend on many factors & differ across flights, routes, seasons etc.
  - Hotels, rental car agencies etc. are increasingly using similar techniques.
- If consumers can predict price changes they can buy at low price points



E.g. Tiered pricing levels for AA flight LAX-BOS on Jan7th. Low fluctuation.(Lowest economy airfares)



E.g. Rapid pricing fluctuations in the lead-up to the new year for AA flight LAX-BOS on Jan 2nd. (Lowest economy airfares)

Airfare Price Prediction Example*

- A data collection agent runs every few hours to screen-scrape the websites of selected airlines and extract pricing information
- Price observations recorded every 3hours over a 41 day period -generating over 12,000 price observations:
  - Flight number
  - Number of hours until departure
  - Current price
  - Airline
  - Route (LAX-BOS or SEA-IAD)
  - Class label = buy / wait
  - Did not have access to other key variables such as number of unsold seats on a flight, whether an airline is running a promotion, or seasonal variables

**Conducted at University of Washington & University of Southern California, 2003*
See http://www.isi.edu/integration/papers/etzioni03-kdd.pdf

Airfare Prediction Example: Learning Algs

Four learning methods were used:

1. Rule-Based Learning (using the *Ripper* algorithm)

IF hours-before-takeoff >= 252 AND price >= 2223
AND route = LAX-BOS THEN *wait*

IF airline = United AND price >= 360
AND hours-before-takeoff >= 438 THEN *wait*

2. Time Series Prediction
   - Predict price using weighted average of the prices on the previous 7 days

$$\frac{\sum_{i=1}^{k} \alpha(i) p_{t-k+i}}{\sum_{i=1}^{k} \alpha(i)}$$

Where α is an increasing function of time (i) – chosen to be simple linear function by trial & error

3. Reinforcement Learning
4. Ensemble of the previous three methods

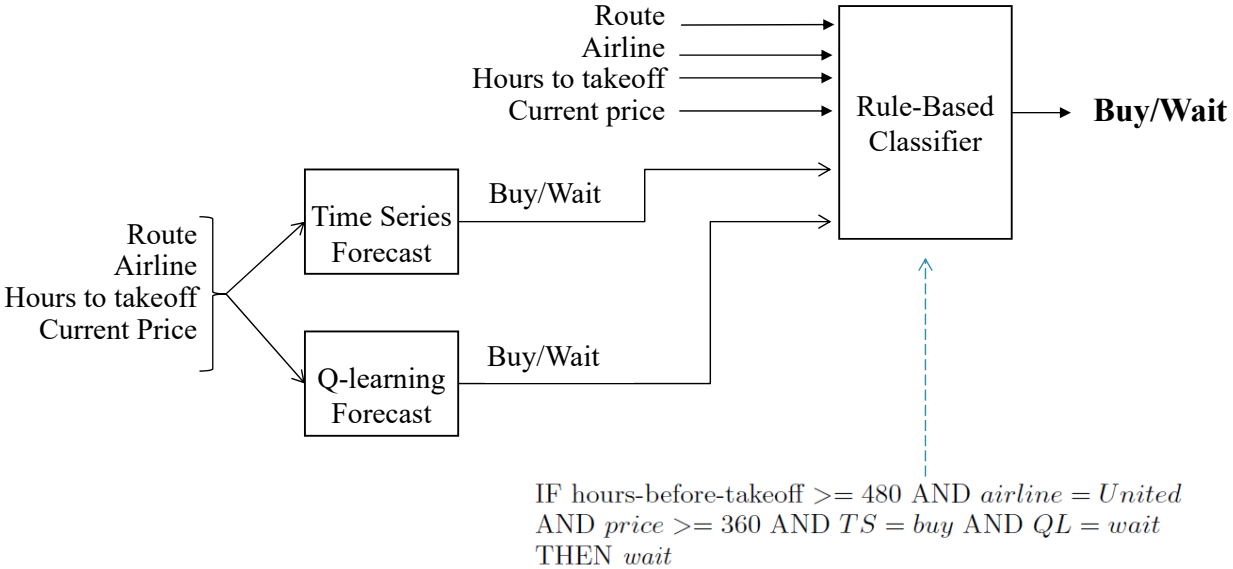
Airfare Prediction : Ensemble Learning

- Use an ensemble (called Hamlet) of the three individual methods
- For each training record
  - Execute the Time series and Q-Learning models to get
    - QL = the output of the Q-Learning prediction model (the QL prediction)
    - TS = the output of the time series prediction model (the TS prediction)
  - Append the QL and TS predictions to the training record
- Learn a new set of prediction rules using Rule learning (Ripper)
- For example:

IF hours-before-takeoff >= 480 AND $airline = United$
AND $price$ >= 360 AND $TS = buy$ AND $QL = wait$
THEN $wait$

Airfare Prediction : Ensemble Learning

- The Experts have same skills (make Buy/Wait decision) but sometimes one is better than the other! The arbitrator helps decide which to use and when



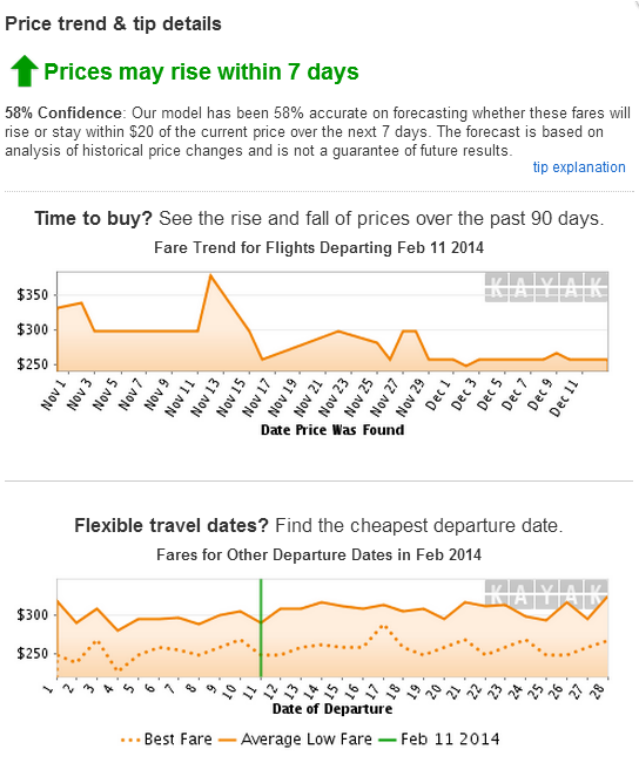
Aside: FareCast.com

- Launched in 2007, acquired by Microsoft a year later (put into Bing Travel, but now retired)
- At launch it had collected 175 billion airfare observations which were fed into models to predict future airfare price movements. Key US routes were predicted only. Now covers US hotel rooms and airfares.
- Accuracy?
  - The original predictions came with a statement that an independent audit by consulting firm Navigant found the predictions accurate 75% of the time on average
  - Independent bloggers found it only 50% accuracy, e.g. this blog from smartertravel.com in 2007

Route	Prediction	Lowest initial fare	Lowest fare after 7 days	Right or wrong?
New York (JFK) to Los Angeles	Fare will drop or stay the same	$339	$359	Wrong
Chicago (O'Hare) to Orlando	Fare will drop by $50	$278	$323	Wrong
San Francisco to Honolulu	Fare will stay the same	$298	$298	Right
Dallas (Ft. Worth) to Boston	Fare will rise by $50	$308	$179	Wrong
Washington, D.C. (National), to New Orleans	Fare will rise or stay the same	$191	$191	Right
Los Angeles to Las Vegas	Fare will stay the same	$98	$98	Right

Aside: Kayak.com

- Airfare price forecasts were launched in Jan'13 for US and UK
- Includes a prediction of whether the prices for your search query will rise or fall in the next week.



Summary

- Ensembles
  - Using multiple models to reduce variance and increase accuracy
  - Usually work by averaging across models
  - Works best if models don't agree with each other (need model variance)
  - Usually refers to multiple models of same type (terminology only)
  - Bagging & Boosting most popular generic methods for building
  - Random Forests increasingly popular
- Multiple Classifier Systems
  - Combining a smaller number of different model types
  - Can also be thought of as Ensembles (by SPSS Modeler)
  - Allows for other model combination methods apart from averaging