Introduction to Predictive Modelling: A Machine Learning Perspective

Elio Amicarelli
Warwick Q-Step
Quantitative Methods Spring Camp 2017

March 21, 2017

Explanation and prediction

"Explanation and prediction have the same logical structure"

— Hampel & Oppenheim (1948)

"It becomes pertinent to investigate the possibilities of predictive procedures autonomous of those used for explanation"

— Helmer & Rescher (1959)

"Theories of social and human behavior address themselves to two distinct goals of science: (1)prediction and (2)understanding"

— Dubin (1969)

Source: Shmueli (2010)

What is predictive modelling?

"The practice of predictive modeling defines the process of developing a model in a way that we can understand and quantify the model's prediction accuracy on future, yet-to-be-seen data." (Kuhn & Johnson APM)

There's gonna be one... NOW! No,... Now! Okay, maybeee... NOW! Alright, it's gonna be... Now! Okayyy... Now!
Another long day down at the Bureau of Earthquake Prediction

Machine Learning

Methods explicitly developed to efficiently deal with predictive tasks and pattern recognition. Core methodological idea: design an algorithm allowing *machines* to *learn* to perform a task from experience.

- Supervised learning $\rightarrow$ Focus of this workshop
- Unsupervised learning
- Semi-supervised learning
- Reinforcement learning

Also designed to meet challenges of modern data analysis (e.g large p, large N, $p > N$, sparsity, violation of parametric assumptions etc...)

Aim of this workshop

" We divided the data into an in-sample **training set** that spanned from 1945 to 2006 (493 observations) and an out-of-sample **test set** that spanned from 2007 to 2012. Bayesian additive **regression trees** (BARTs) were used to analyze the data. BART uses the ensemble regression tree methods popular in **machine learning prediction models**, but does so within a formal probability framework, which makes it generally **more stable**. Our experience was that BART required far less **tuning** and produced more consistent results than other **ensemble methods**, like **boosting** or **random forests**. Tuning on the training set was done through grid-search **cross-validation**." (Kennedy et al. 2017)

Setting the stage: terminology and predictive task

$$Y = f(X) + \epsilon \tag{1}$$

$$\hat{Y} = \hat{f}(X) \tag{2}$$

Y: target, output, response, outcome
X: predictors, input variables, features space
f: function (rule) mapping X to Y

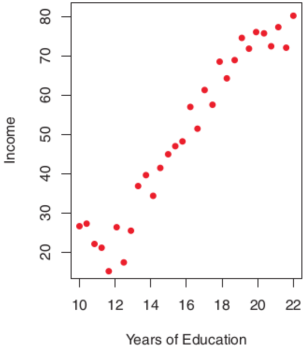
From a procedural point of view, our task is the following:

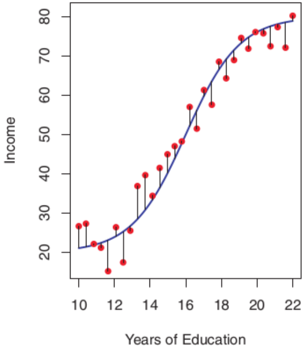
i) To **train** a model on observed input/output pairs $(x_1, y_1), (x_2, y_2), (x_3, y_3)$ so that

ii) it effectively **learns** a rule $\hat{f}$ connecting X and Y which allows us

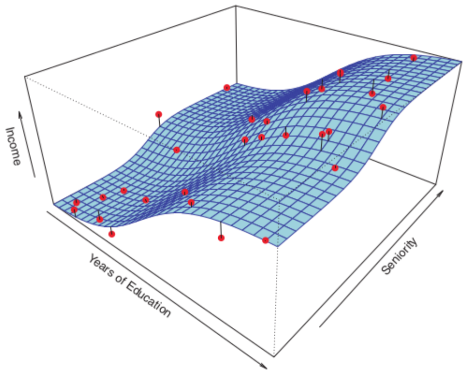
iii) to accurately predict the value of y_4 from a new **test input** x_4

Setting the stage: terminology and predictive task





Setting the stage: terminology and predictive task



The two main phases of predictive modelling

Key insights:

- There are two **separate** phases in the predictive modelling pipeline: a **training** phase and a **testing** phase.
- To mimic a situation where we predict future outcomes, we split the main data set in two sets. The **train set** will be used during the training phase, while the **test set** will be used for testing purposes.
- It is important to evaluate your **models' performances** in each of these phases but with **two distinct goals** in mind.
- **Golden rule of predictive modelling: Never use test data to train your models**

Performance assessment: goals

It is important to evaluate your **models' performances** in each of these phases but with **two distinct goals** in mind.

Assessing model performances in the training phase:

- Select the best $\hat{f}(X)$ among different candidates
- Select the right level of **model's complexity**

Assessing model performances in the testing phase:

- Evaluate the predictive performances of the final model on new data

Performance assessment: methods

Remember:

$$Y = f(X) + \epsilon \tag{3}$$

$$\hat{Y} = \hat{f}(X) \tag{4}$$

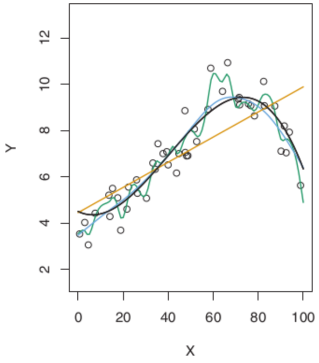
General idea:

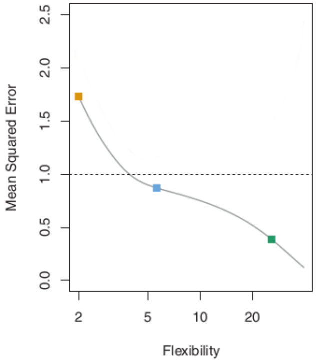
$$Err(\hat{f}(X)) = \mathbb{E}[\ell(Y, \hat{f}(X))] \tag{5}$$

Important example:

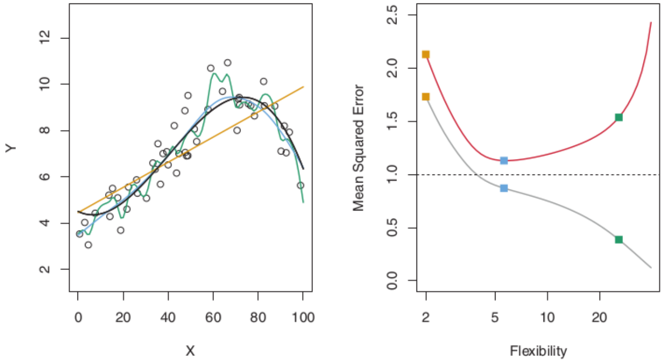
$$MSE = \mathbb{E}[(\hat{Y} - Y)^2] = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \tag{6}$$

Assessment during training



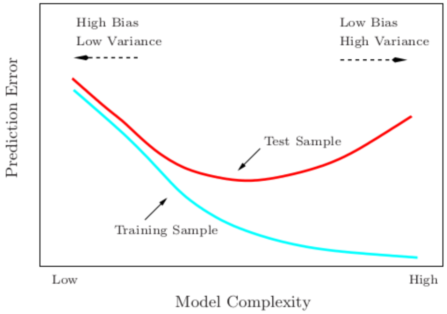


Assessment during training



Assessment during training: the challenge

- The prediction error on the train set gets smaller as model complexity increases because a more complex model is better able to exactly remember and reproduce specific training data.
- However, the generalization error (i.e. the prediction error on the test set) also decreases initially as model complexity increases, but beyond a certain point it starts to increase.



Assessment during training: the challenge

An overly complex model has low bias and high variance because is capturing both the signal and the noise in the data. In this case we say the model is **overfitting** the data.

An overly simplistic model has high bias and low variance because is not capturing neither the signal nor the noise in the data. In this case we say the model is **underfitting** the data.

Both underfitting and overfitting lead to suboptimal predictive performances. For this reason we want to **tune** our model's complexity in order to hit the sweet spot along the **bias-variance tradeoff**.

Time to run some code!

Bias-variance decomposition

Assume $Y = f(X) + \epsilon$ with $\mathbb{E}[\epsilon] = 0$ and $Var(\epsilon) = \sigma^2$. Under a quadratic loss function we can show the following:

$$\begin{aligned}
Err(\hat{f}) &= \mathbb{E}[(Y - \hat{f})^2] \\
&= \mathbb{E}[Y^2] + \mathbb{E}[\hat{f}^2] - \mathbb{E}[2Y\hat{f}] \\
&= Var[Y] + \mathbb{E}[Y]^2 + Var[\hat{f}] + \mathbb{E}[\hat{f}]^2 - 2f\mathbb{E}[\hat{f}] \\
&= Var[Y] + Var[\hat{f}] + (f^2 - 2f\mathbb{E}[\hat{f}] + \mathbb{E}[\hat{f}]^2) \\
&= Var[Y] + Var[\hat{f}] + (f - \mathbb{E}[\hat{f}])^2 \\
&= \sigma^2 + \mathbb{E}[(\hat{f} - \mathbb{E}[\hat{f}])^2] + (f - \mathbb{E}[\hat{f}])^2
\end{aligned}$$

Assessment during training: the challenge

We are left with a puzzle:

- We are interested in selecting the best $\hat{f}^*$ to **minimize the generalization error** (e.g. MSE_{test})
- But we are **not allowed to use the test set** to choose among different $\hat{f}$s. All we can use to select $\hat{f}^*$ is our train set!
- And unfortunately we know that it is dangerous to select $\hat{f}$ based on the training error (e.g. MSE_{train}) because of the risk of **overfitting**

How can we solve this problem?

Assessment during training: the solution

To find $\hat{f}^*$ using only our training data we are going to use **k-fold cross-validation**. This will allow us to estimate the generalization error without touching our test set thus respecting the golden rule.

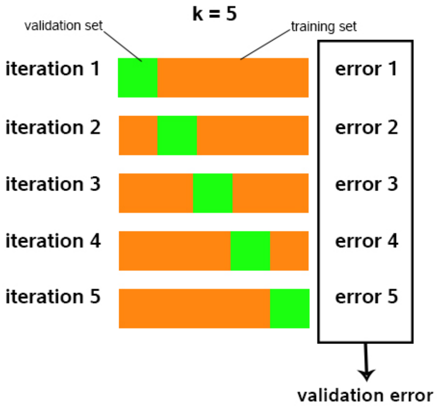
k-fold cross-validation

K-fold cross-validation (CV) is a model selection procedure which involves:

- splitting the train data into k folds (e.g. 5 folds)
- leaving out one of these folds (e.g. fold number 1) and training the model on the remaining folds (e.g. using folds from 2 to 5 together as new training set)
- predicting the response values in the fold not used to train the model and calculate the prediction error
- repeating the procedure until each fold has been excluded from the training data once

In this scheme, each fold is used once as a **validation set** and $k - 1$ times as part of a larger training set. For each $\hat{f}$ now we have k prediction errors.The **validation error** for a given $\hat{f}$ can be calculated as the average of these k prediction errors.

k-fold cross-validation



Assessment during testing

The selection of a suitable model via cross-validation concludes the training phase. Now that we have finally settled for our $\hat{f}^*$ the next step is very straightforward and way less laborious than the training stage. In the testing phase we feed the test set inputs X_{test} into $\hat{f}^*$ in order to obtain predictions for the test outcome:

$$\hat{Y}_{test} = \hat{f}^*(X_{test}) \tag{7}$$

Assessment during testing

Then, we compare our predictions with the actual test outcomes in order to obtain a **test error**. You should be consistent across training and testing phase regarding the measure adopted to assess the performances. If you adopted the mean squared error during the training, then you should use it for testing purposes too:

$$MSE_{test} = \mathbb{E}[(\hat{Y}_{test} - Y_{test})^2] \tag{8}$$

Assessment during testing

The test error gives us:

- An estimate of how well our model performs in predicting unseen outcomes
- A common ground to compare the predictive performances of different models

Tracking learners

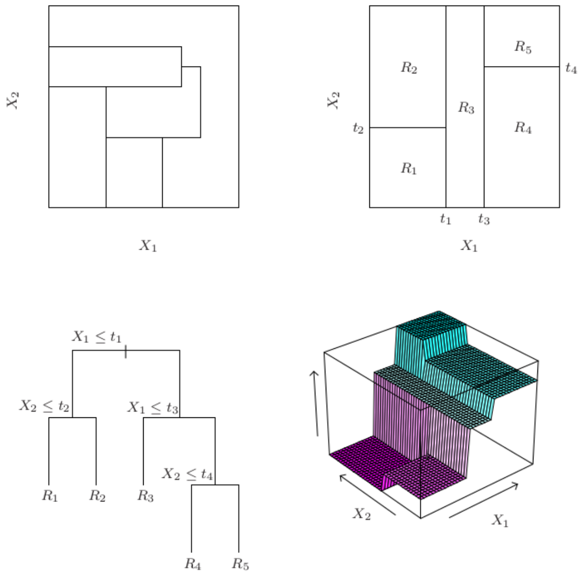
" We divided the data into an in-sample **training set** that spanned from 1945 to 2006 (493 observations) and an out-of-sample **test set** that spanned from 2007 to 2012. Bayesian additive **regression trees** (BARTs) were used to analyze the data. BART uses the ensemble regression tree methods popular in **machine learning prediction models**, but does so within a formal probability framework, which makes it generally **more stable**. Our experience was that BART required far less **tuning** and produced more consistent results than other **ensemble methods**, like **boosting** or **random forests**. Tuning on the training set was done through grid-search **cross-validation**." (Kennedy et al. 2017)

Classification and Regression Trees (CART)

The key points of our discussion:

- What CART are and how they can be used to make predictions
- How to grow classification trees
- How to evaluate a classifier's predictive performances
- Bias-variance considerations in CART

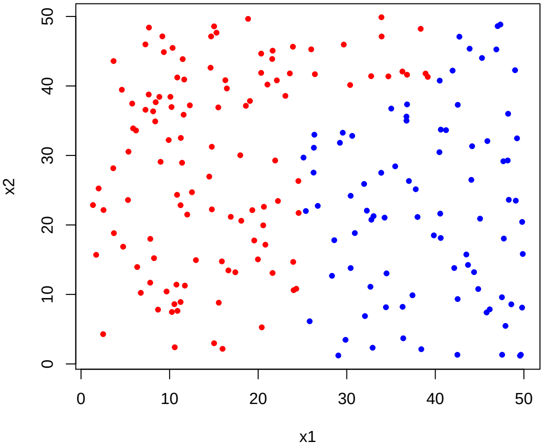
Anatomy of a decision tree



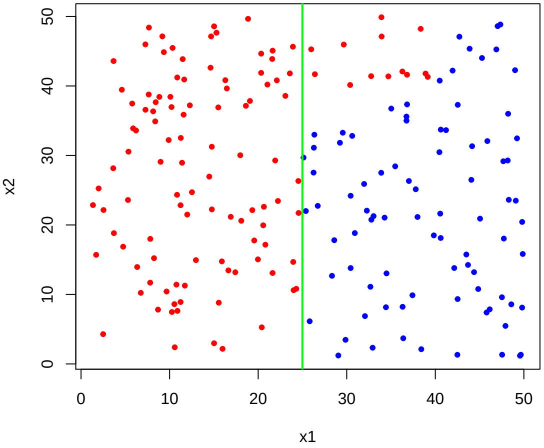
Classification and Regression Trees (CART)

- Given a training set, a tree algorithm recursively creates binary non-overlapping partitions on the features space (X) in order to minimize the dissimilarity among the outcome values (Y) falling in each partition.
- The prediction corresponding to a given region of the partitioned space is usually obtained by applying a simple aggregating rule on the training outcomes falling in that region (e.g. the average for regression tasks and the majority class for classification).

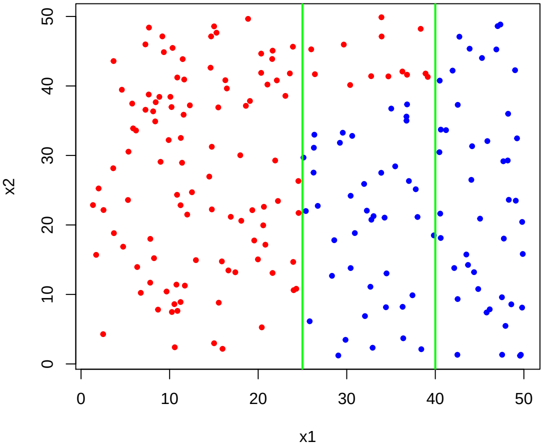
Building a classification tree: example



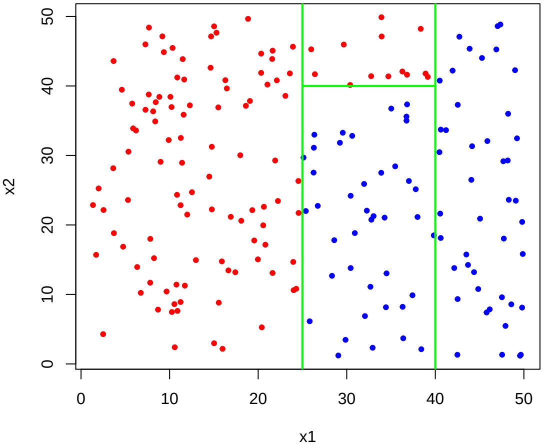
Building a classification tree: example



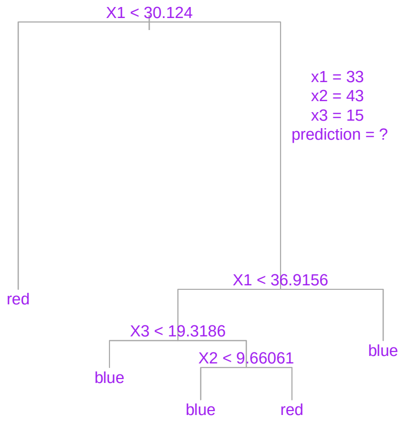
Building a classification tree: example



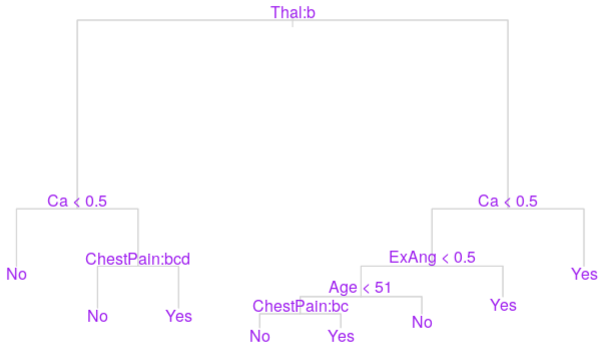
Building a classification tree: example



Predictions with classification tree: example



Predictions with classification tree: interpretation



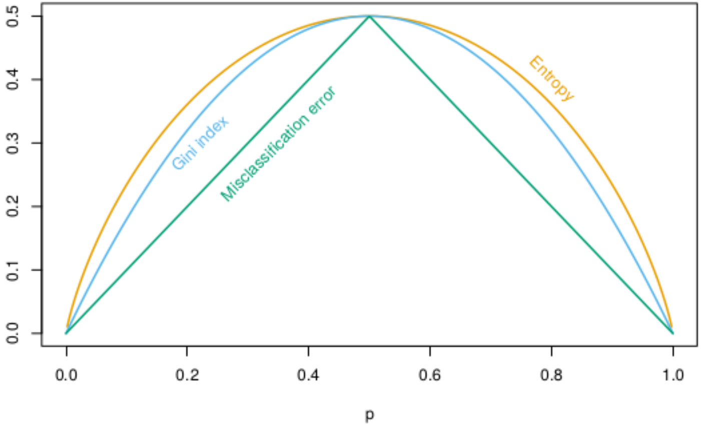
Building a classification tree: impurity

$$CE = 1 - \max(p_j) \tag{9}$$

$$Gini = 1 - \sum_{j=1}^{J} p_j \tag{10}$$

$$Entropy = -\sum_{j=1}^{J} p_j log_2 p_j \tag{11}$$

Building a classification tree: impurity



Building a classification tree: algorithm

Tree algorithm:

1. At each stage, select a variable from the features space X (e.g X_1) and a value for that variable (e.g. v_1), so that the split performed on (X_1, v_1) **maximizes the reduction over the current node's impurity**.
2. Repeat Step 1 until each terminal node has fewer than some minimum number of observations
3. Apply cross-validation to select the best level of tree-depth (pruning the tree)
4. In order to produce predictions given a new set of X values
   - Use the tree structure to identify the region corresponding to the given X values
   - The prediction for the response is the majority class of the identified region

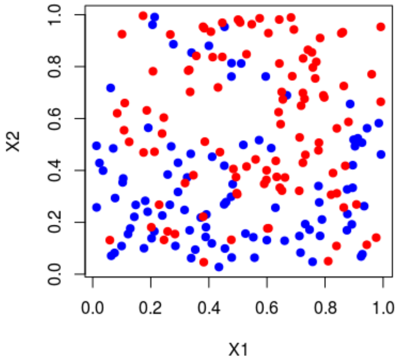
Building a classification tree: information gain

Let $i(t)$ be the impurity score of a given node according with one of the impurity measures considered before. The next split will be performed on the pair (x, v) which maximizes the **information gain** defined as:

$$\delta i(t) = i(t) - p_L i(t_L) - p_R i(t_R) \tag{12}$$

Where p_L and p_R are the proportions of data in t that go to the left and right region respectively, while $i(t_L)$ and $i(t_R)$ are the impurity scores of these regions.

Building a classification tree: impurity



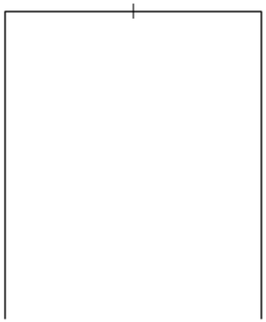
For a nice simulation on tree partitioning via Gini index see http://freakonometrics.hypotheses.org/1279 (source of this image)

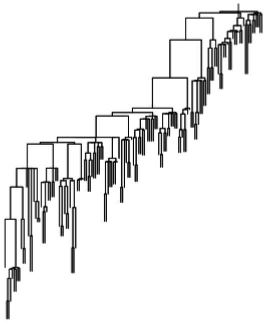
CART bias-variance considerations

When do overfitting and underfitting occur in CARTs?

CART bias-variance considerations

When do overfitting and underfitting occur in CARTs?





To conclude

CART pros and cons:

- Easily interpreted
- Fast
- Effective with non-linear decision boundaries
- Predictive accuracy inferior to more sophisticated models
- Unstable

Addendum: Evaluating classifiers' predictive performances

Confusion matrices are very intuitive and useful tools that can be used to evaluate the performances of any **classifier**. A confusion matrix allows you to look at different aspects of a classifier's performances and to compare different classifiers in order to establish which one is a better predictive model. A confusion matrix is simply a cross-tabulation between the model's predictions ($\hat{Y}$) and the actual response values (Y).

n=192	**Predicted: 0**	**Predicted: 1**
Actual: 0	118	12
Actual: 1	47	15

Addendum: Evaluating classifiers' predictive performances

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{13}$$

$$TPR = \frac{TP}{TP + FN} \tag{14}$$

$$PPV = \frac{TP}{TP + FP} \tag{15}$$

$$TNR = \frac{TN}{TN + FP} \tag{16}$$

$$FPR = \frac{FP}{TP + FN} \tag{17}$$

$$FNR = \frac{FN}{TP + FN} \tag{18}$$

Ensemble learning

- General concept: The combination of different models outperforms their individual predictive performances
- A long-known phenomenon: Condorcet's jury theorem (1785), Francis Galton's wisdom of the crowds (1907)
- Core area of Machine Learning: ensemble models dominate (almost?) all predictive competitions

Bootstrap aggregation

Initially proposed by Breiman (1996), **bootstrap aggregation (bagging)** is a general-purpose ensemble procedure for **reducing the variance** of a predictive model:

- Consider a training set $\mathcal{L}$
- Produce M bootstrap samples $\mathcal{L}_m (m = 1, 2, ..., M)$
- Train a model on each $\mathcal{L}_m$ thus obtaining M variants of $\hat{f}_m(X_{\mathcal{L}_m})$
- Given a new set of X, produce predictions from each $\hat{f}_m(X_{\mathcal{L}_m})$
- Combine the predictions of all models

Bootstrap aggregation: performances

- The aggregated predictions have lower variability than the prediction that might be generated by an individual model
- Analysis across several dataset have shown that reduction in misclassification rates is considerable
- Estimation of generalization error is almost built-in thanks to **out-of-bag** observations
- When bagging is applied to CART trees are grown deeply!

Random Forests

Breiman strikes again in 2001 introducing Random Forests. Usually applied using CART, Random Forests differ from bagging in just one step:

- Consider a training set $\mathcal{L}$
- Produce M bootstrap samples $\mathcal{L}_m(m = 1, 2, ..., M)$
- Fit a deep tree on each $\mathcal{L}_m$, but at each node use only n variables sampled from X, thus obtaining M variants of $\hat{f}_m(X_{\mathcal{L}_m})$
- Given a new set of X, produce predictions from each $\hat{f}_m(X_{\mathcal{L}_m})$
- Combine the predictions of all models

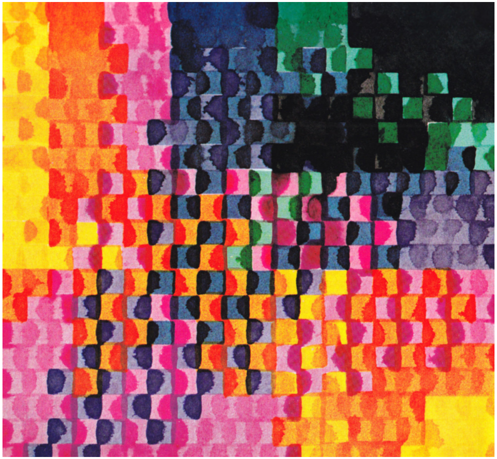
Random Forests: performances

- Individual models in Random Forests are more decorrelated than in bagging thus further reducing the variance of the predictions and enhancing model's performances
- Random Forests usually outperform bagging but tend to be inferior to other ensemble techniques (e.g. gradient boosting)
- Nice thing is that Random Forests have just two parameters we need to tune
- Estimation of generalization error is almost built-in thanks to **out-of-bag** observations
- We can combine the decreases in impurity deriving from splits performed on specific variables in order to evaluate **variables' importance**

Tracking learners

“ We divided the data into an in-sample **training set** that spanned from 1945 to 2006 (493 observations) and an out-of-sample **test set** that spanned from 2007 to 2012. Bayesian additive **regression trees** (BARTs) were used to analyze the data. BART uses the ensemble regression tree methods popular in **machine learning prediction models**, but does so within a formal probability framework, which makes it generally **more stable**. Our experience was that BART required far less **tuning** and produced more consistent results than other **ensemble methods**, like **boosting** or **random forests**. Tuning on the training set was done through grid-search **cross-validation**.” (Kennedy et al. 2017)

Questions?



Frieder Nake, Matrix multiplication (1967)

Where to go from here

Useful books:

- James, Gareth, et al. *An introduction to statistical learning*. New York: springer, 2013. (ISL)
- Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Springer, Berlin: Springer series in statistics, 2001. (ESL)
- Kuhn, Max, and Kjell Johnson. *Applied predictive modeling*. New York: Springer, 2013. (APM)
- Barber, David. *Bayesian reasoning and machine learning*. Cambridge University Press, 2012.

Learn by doing:

- Online competitions
- Hackathons
- Research