Introduction to Data Science

BRIAN D'ALESSANDRO

ADJUNCT PROFESSOR, NYU

FALL 2016

MODEL SELECTION

GOALS OF EVALUATION

The goal of evaluation in model building can be put quite simply: achieve the ***best generalization performance while avoiding overfitting***.

Remember, in ERM we seek a function f(X) that minimizes the training error (or risk) on our training data.

$$R_{train} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{L}(f(x_i^{train}), y_i^{train})$$

WE ALWAYS want to measure the error on a holdout, or test set, too.

$$R_{test} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{L}(f(x_i^{test}), y_i^{test})$$

We can only be certain our models generalize if we do a correct holdout evaluation. Theory helps us design good algorithms with strong generalization and convergence results, but empirical hold-out testing is always necessary.

REMEMBER

Expected Risk != Empirical Risk

Training error is our empirical risk and test set error is our best approximation of expected risk.

MODEL SELECTION

When we model, we have many choices. We use holdout evaluation methodologies to optimize the following:

- ***Algorithm Selection***
  (i.e., RF vs. Decision Tree vs. Logistic Regression vs. SVM)

- ***Feature Selection***

- ***Hyper-parameter Selection***, i.e.,
  - "k" in k-NN
  - "C" in SVM
  - MaxDepth, MinLeafSize in Decision Trees
  - Regularization Strength

The optimal configuration of these elements depends on the goals of the problem, which define an appropriate evaluation metric.

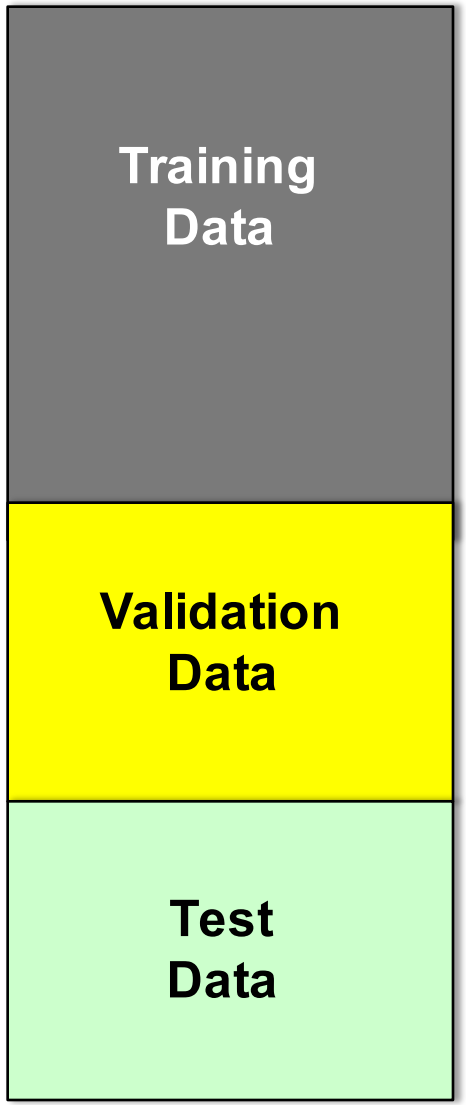
QUICK HYPER-PARAMETER REVIEW

So far we have learned the following algorithms and their associated complexity parameters.

- ***Logistic Regression***
  - C (regularization weight)
  - L1/L2 (regularization strategy)
- ***Support Vector Machine***
  - C (regularization weight)
  - Kernel (and its associated hyperparameters)
- ***Decision Trees***
  - MaxDepth
  - MinLeafSize
  - MinSplitSize

BASIC DESIGN

When doing any sort of model selection, one usually begins by creating 3 splits of the data.



Training: the training data is used to find the optimal function given the model structure (i.e., fixed algorithm, feature set, hyper-parameters).

Validation: the validation data is used to evaluate the loss/risk for a given model configuration. The configuration with the best loss/risk is selected as the final model

Test: test data is not used for any parameter or model selection. It is only used as a generalization measure.

EXAMPLE MODEL SELECTION ROUTINE

Define: Training Data
Define: Validation Data
Define: Test Data

1. For each configuration c in the set C=[Algorithm x Feature Set x Hyper-parameter]:

- Find the function $\hat{f}_c$ such that: $\hat{f}_c = \underset{f \in \mathbb{F}}{\text{argmin}}\ R^{train}$
- With $\hat{f}_c$ estimated, get the validation loss: $R_c^{val} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{L}(\hat{f}_c(x_i^{val}), y_i^{val})$

2. Choose the optimal configuration c_{opt} such that: $c_{opt} = \underset{c \in C}{\text{argmin(max)}}\ R_c^{val}$

3. Define: NewTrain = Train + Validation data

4. Find the function $\hat{f}$ such that: $\hat{f} = \underset{f \in \mathbb{F}}{\text{argmin}}\ R^{NewTrain}$

5. Estimate the test loss as: $R^{test} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{L}(\hat{f}(x_i^{test}), y_i^{test})$

SOME NOTES ON VALIDATION RISK

The validation loss metric does not have to be the same as the training loss.

- Sometimes we need a loss metric for an application that is not very easy or even possible to directly minimize.
- I.e., we use logistic-loss to find *f* but we really need a loss metric that enables optimal ranking (such as AUC)
- We might want to minimize a training loss metric, but maximize the validation metric (again logistic-loss vs. AUC)

$f \in \mathbb{F}$

- With $\hat{f}_c$ estimated, get the validation loss: $R_c^{val} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{L}(\hat{f}_c(x_i^{val}), y_i^{val})$

2. Choose the optimal configuration c_{opt} such that: $c_{opt} = \text{argmin(max)}\ R_c^{val}$

CROSS-VALIDATION

Sometimes we don't have enough data to do a single train/validation/test split.

We can "recycle" data by using k-fold cross validation as our validation scheme.

EXAMPLE X-VALIDATION SCHEME

Define: Training Data
Define: Test Data

1. **For each configuration c in the set C=[Algorithm x Feature Set x Hyper-parameter]:**

   - For each fold k:
     - Find the function $\hat{f}_c^k$ such that: $\hat{f}_c^k = \underset{f \in \mathbb{F}}{\text{argmin}}\, R^{train-k}$
     - With $\hat{f}_c^k$ estimated, get the validation loss: $R_c^{val-k} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{L}(\hat{f}_c(x_i^{val-k}), y_i^{val-k})$
   - Get the average loss across all folds: $R_c^{val} = \sum_{i=1}^{k} R_c^{val-k}$

2. **Choose the optimal configuration c_{opt} such that:** $c_{opt} = \underset{c \in C}{\text{argmin(max)}}\, R_c^{val}$
3. **Find the function $\hat{f}$ such that:** $\hat{f} = \underset{f \in \mathbb{F}}{\text{argmin}}\, R^{train}$ **using** c_{opt}
4. **Estimate the test loss as:** $R^{test} = \frac{1}{n} \sum_{i=1}^{n} \mathbb{L}(\hat{f}(x_i^{test}), y_i^{test})$

SOME X-VAL NOTES

How should we choose split size or k?

There is no golden rule that defines how big training, validation and test sets should be.

You want each data set to be big enough to reduce the variance of your estimates.

Estimation variance comes in two flavors – variance of the function being fit to the training data and variance of the estimate of the validation/test risk.

Good heuristics are 70%/20%/10% for train/val/test splits and k=5 or 10 for x-validation.

Also, data should be randomly split with no overlap in instances between sets.

EXAMPLE MODEL SELECTION

We have a dataset with 14 features, roughly 10k examples and only 253 positive examples.

Goal

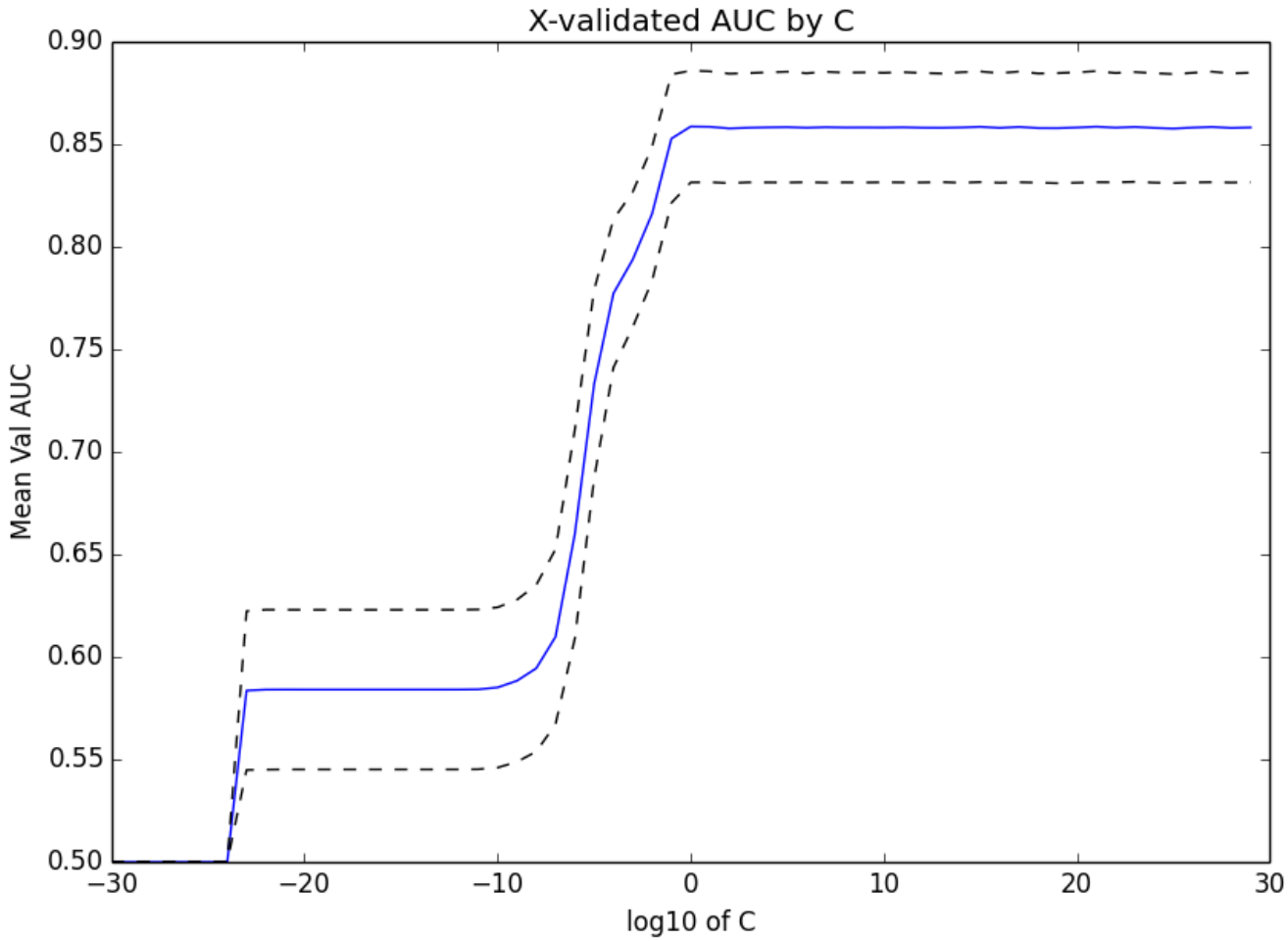
Build a classifier that has good ranking properties.

Solution

1. We'll use Logistic Regression because it is robust in small-sample sizes and imbalanced classes, and also returns a score instead of just label predictions
2. With small data we expect high variance, so we need to use regularization
3. We'll split data into 80/20 train/test and run 10-fold xvalidation on train.
4. We'll use AUC to choose a regularization weight

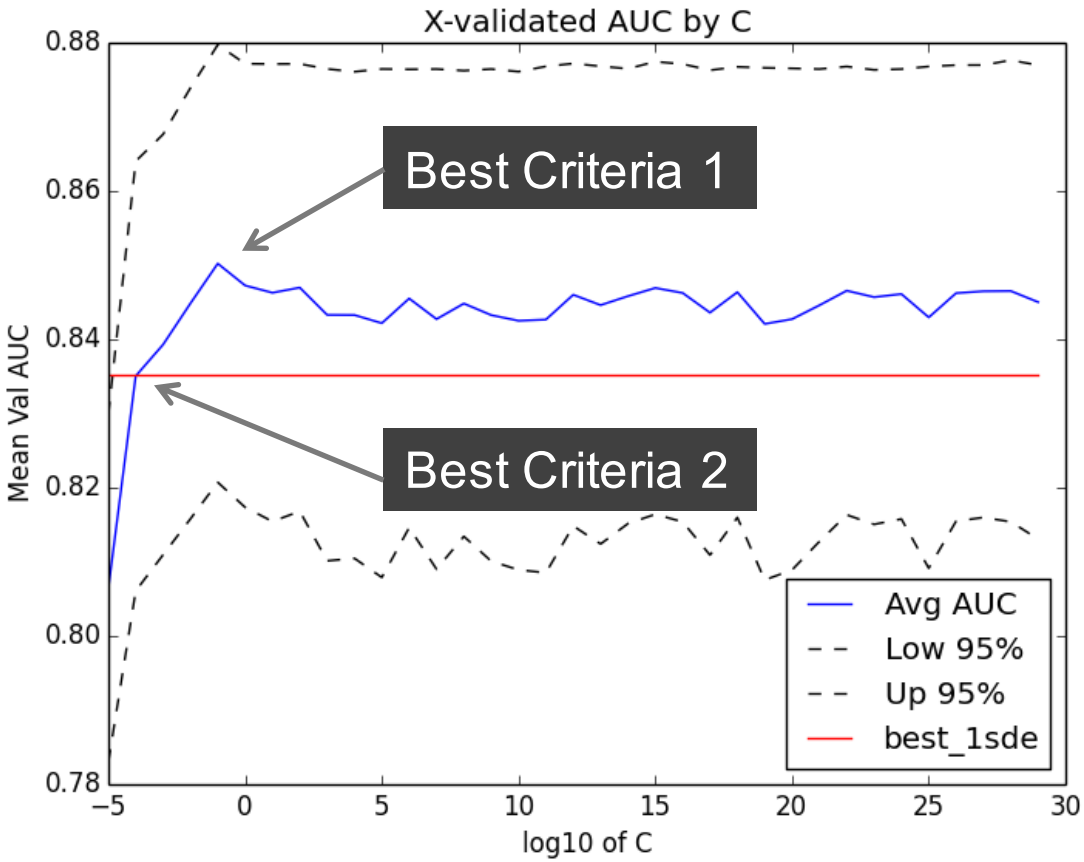
RESULTS OF XVALIDATION

With less regularization we actually do better here (strong signal in the features). But we still see that between 1 to 10^{30} we get nearly the same results. It would be better if we can zoom into the region where performance is better.



RESULTS OF XVALIDATION -ZOOMED

When we zoom in we can see that statistically speaking, everything above C=1 is essentially the same (i.e., strongly overlapping confidence intervals). There are two ways to choose C here.



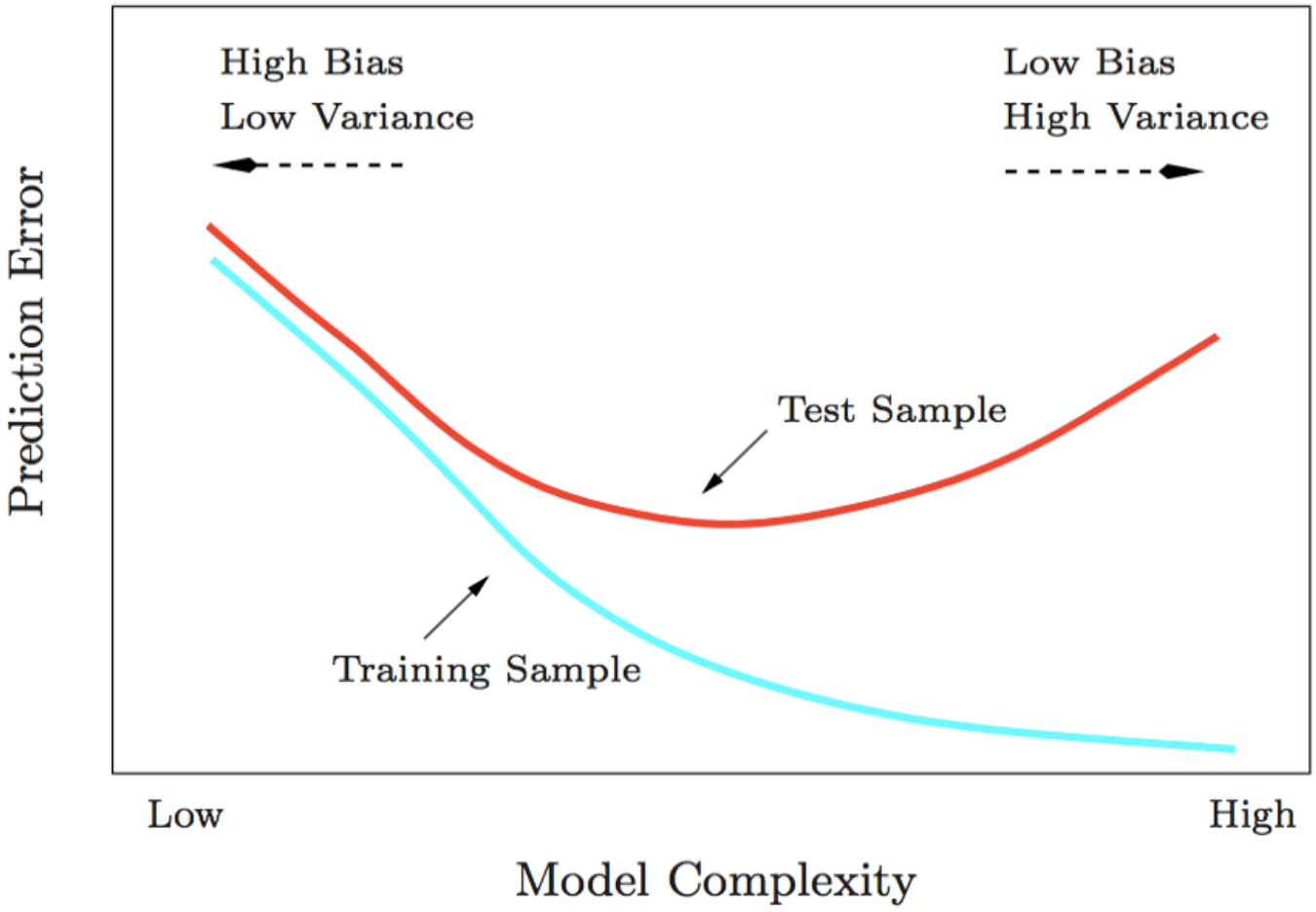
Criteria 1:
Choose option with max(Xval AUC)

Criteria 2:
Find all options where AUC>=max(AUC) – 1stderror

Choose least complex option (highest regularization)

FINDING THE PERFORMANCE SWEEET SPOT

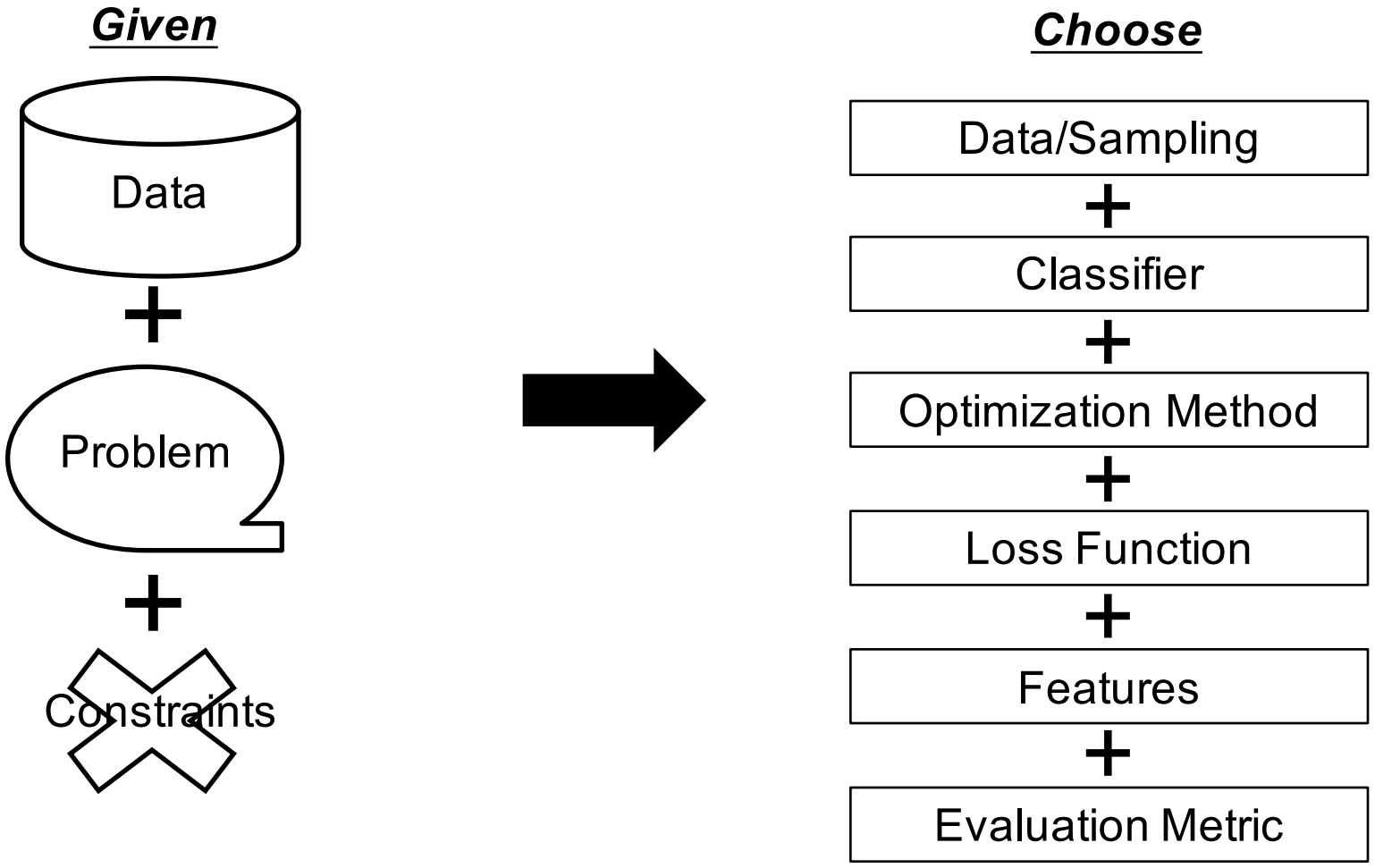
While theory motivates the algorithms we use, finding the optimal bias-variance tradeoff is generally always an empirical process. The routines presented here give us generally our best chance at being successful.



MODEL SELECTION AND SOLUTION ENGINEERING

A COMMON THEME

Few problems have out of the box solutions



The Data Scientist has to navigate these choices

CLASSIFICATION ALGORITHMS

The following is a non-exhaustive list of popular algorithms used in classification problems:

Classic & Simpler Methods

Decision Tree
Naïve Bayes
K- Nearest Neighbors
Linear Hyperplane

Black Box but Powerful Methods

Random Forests
Non-Linear SVM
Neural Networks

We will NOT discuss each of these algorithms in detail in this course, but we will cover the process of how to choose one.

BUT WHICH ONE SHOULD I USE?

If world free of constraints, then (e.g. a data mining competition):

Try them all, choose best performer

Else:

Consider all constraints on your problem.
Choose best performer subject to constraints

TRY THEM ALL???

Train = Training Data
Val = Validation Data

For each Algorithm in <set of all algorithms>:

Build a classifier, $F^A(X)$ using **Train**

Get out-of-sample error of $F^A(X)$ using **Val**

Choose the Algorithm with the best out-of-sample error.

BAKEOFF RULES

1. Training data must always be disjoint from validation data.

2. Use the same training data and validation data for each hypothesis being tested.

3. Given a tie (statistical or exact), choose the simpler model (sometimes this is subjective).

4. Use this methodology for all design decisions (feature selection, hyper-parameter selection, model selection, etc.)

CONSTRAINTS TO CONSIDER

Do you have the right data?

- Production data is often biased
- The data for your problem might not exist (cold start, new product, etc.)
- The right metric/outcome is unmeasurable (i.e., buys orange juice, or consumer happiness)
- The outcome is so rare you barely observe it

Too little/too much data?

- There are few observations but many variables (generally an estimation problem)
- Too much data (generally a computation problem)

CONSTRAINTS TO CONSIDER

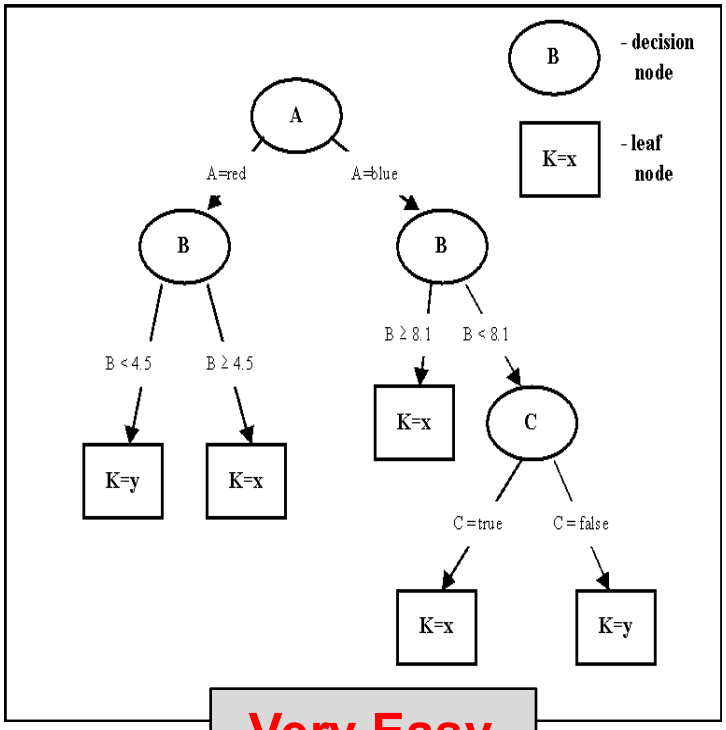
Do you understand the algorithm?

- Your own personal knowledge is a constraint worth admitting to
- You don't have to master every algorithm to be a good data scientist
- Getting the "best-fit" of an algorithm often requires intimate knowledge of said algorithm

CONSTRAINTS TO CONSIDER

Do you need to interpret the model?

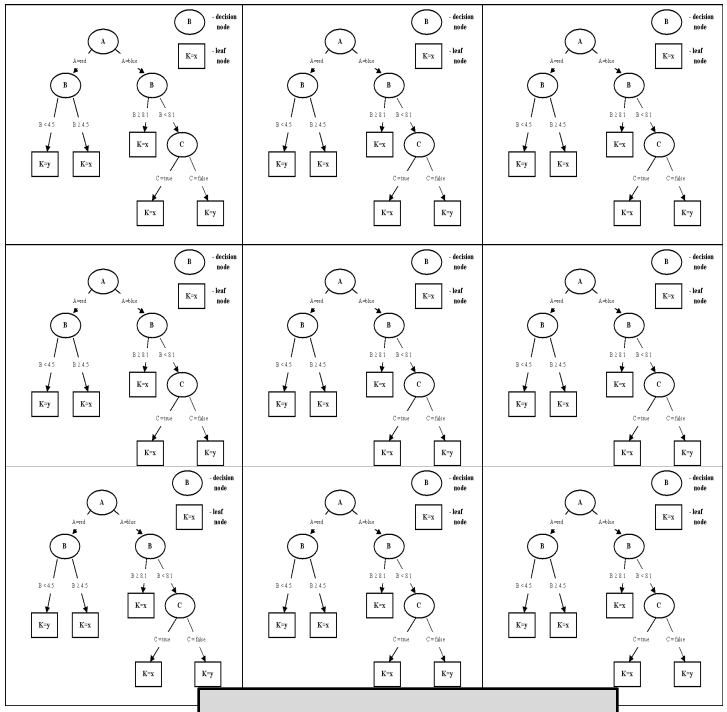
Decision Tree



Very Easy

Vs.

Random Forest

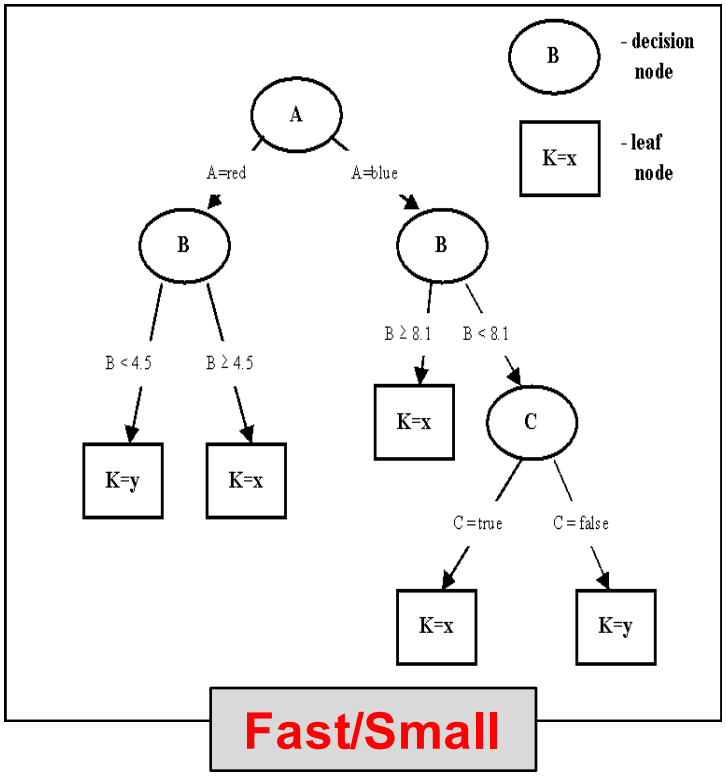


Very Difficult

CONSTRAINTS TO CONSIDER

Does scalability matter (learning time, scoring time, model storage)?

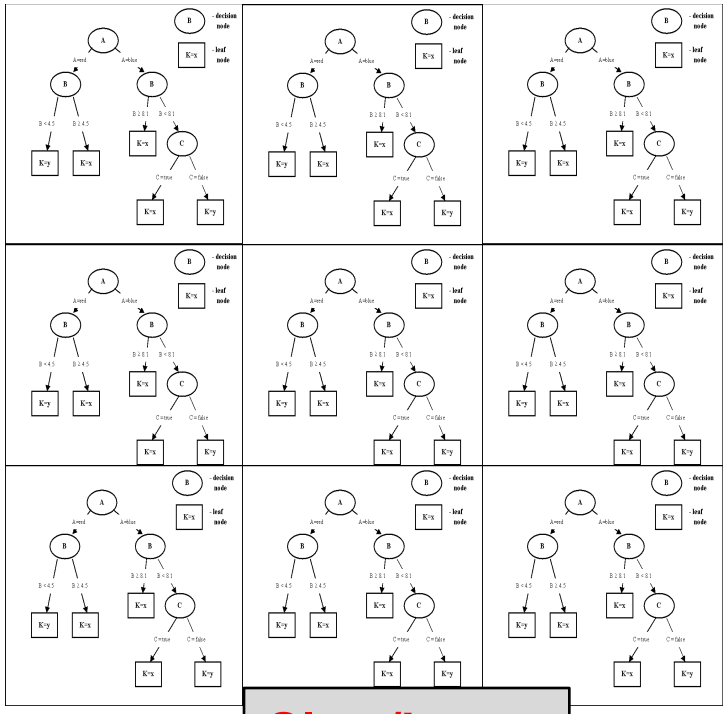
Decision Tree



Fast/Small

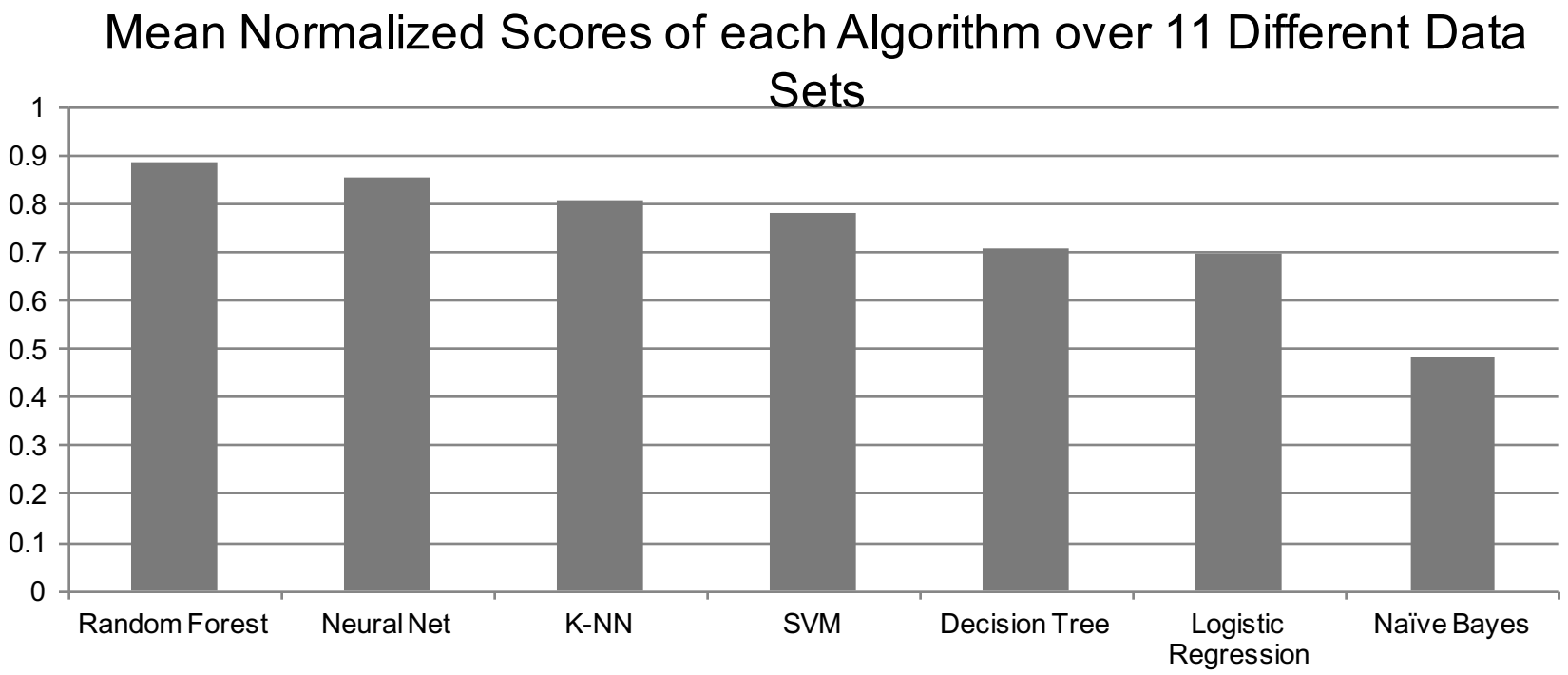
Vs.

Random Forest



Slow/Large

AN EMPIRICAL COMPARISON OF CLASSIFICATION ALGORITHMS



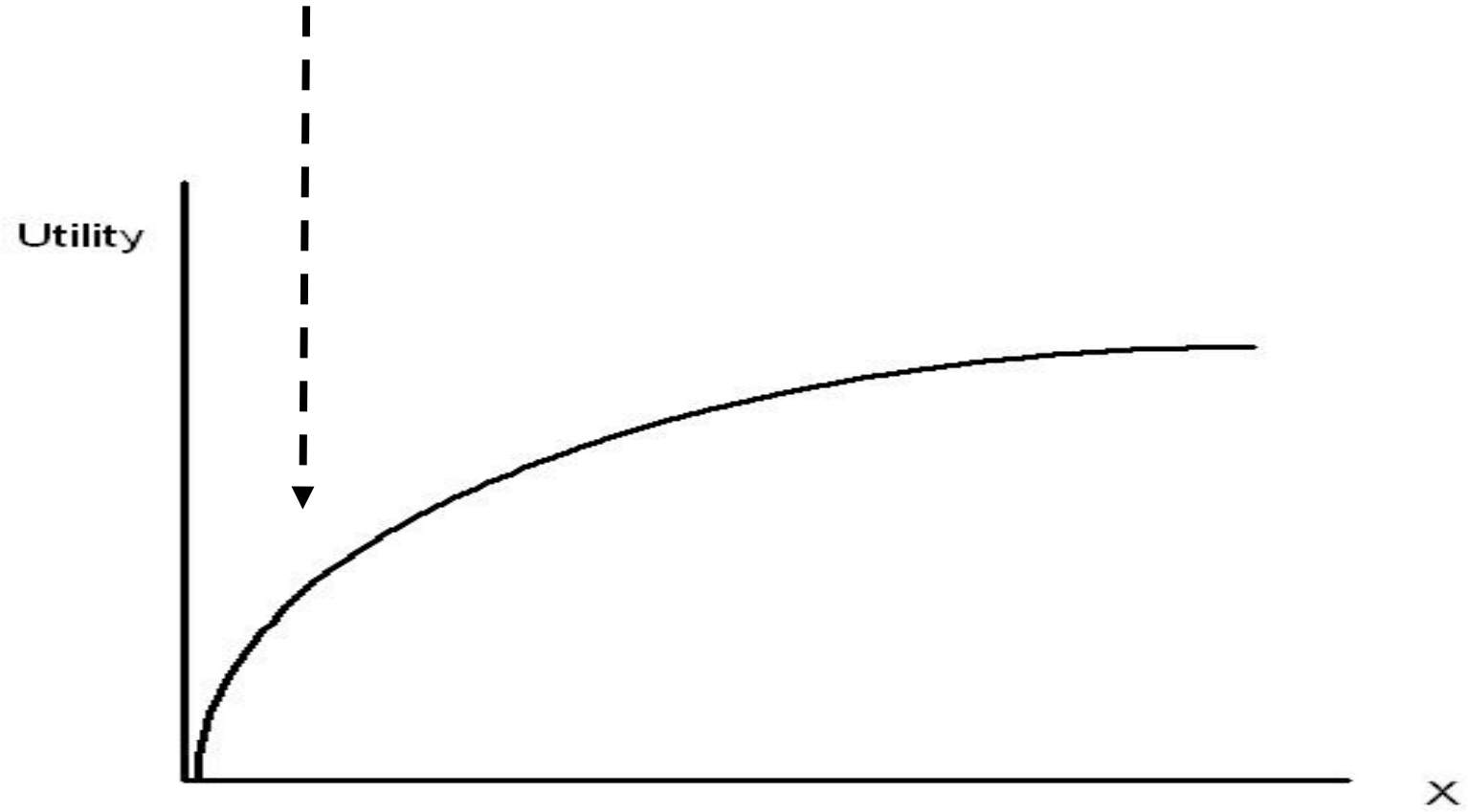
Scalability/Complexity/Interpretability

Performance

Source: An Empirical Comparison of Supervised Learning Algorithms http://www.niculescu-mizil.org/papers/comparison.tr.pdf

ALWAYS BE AGILE: ITERATE

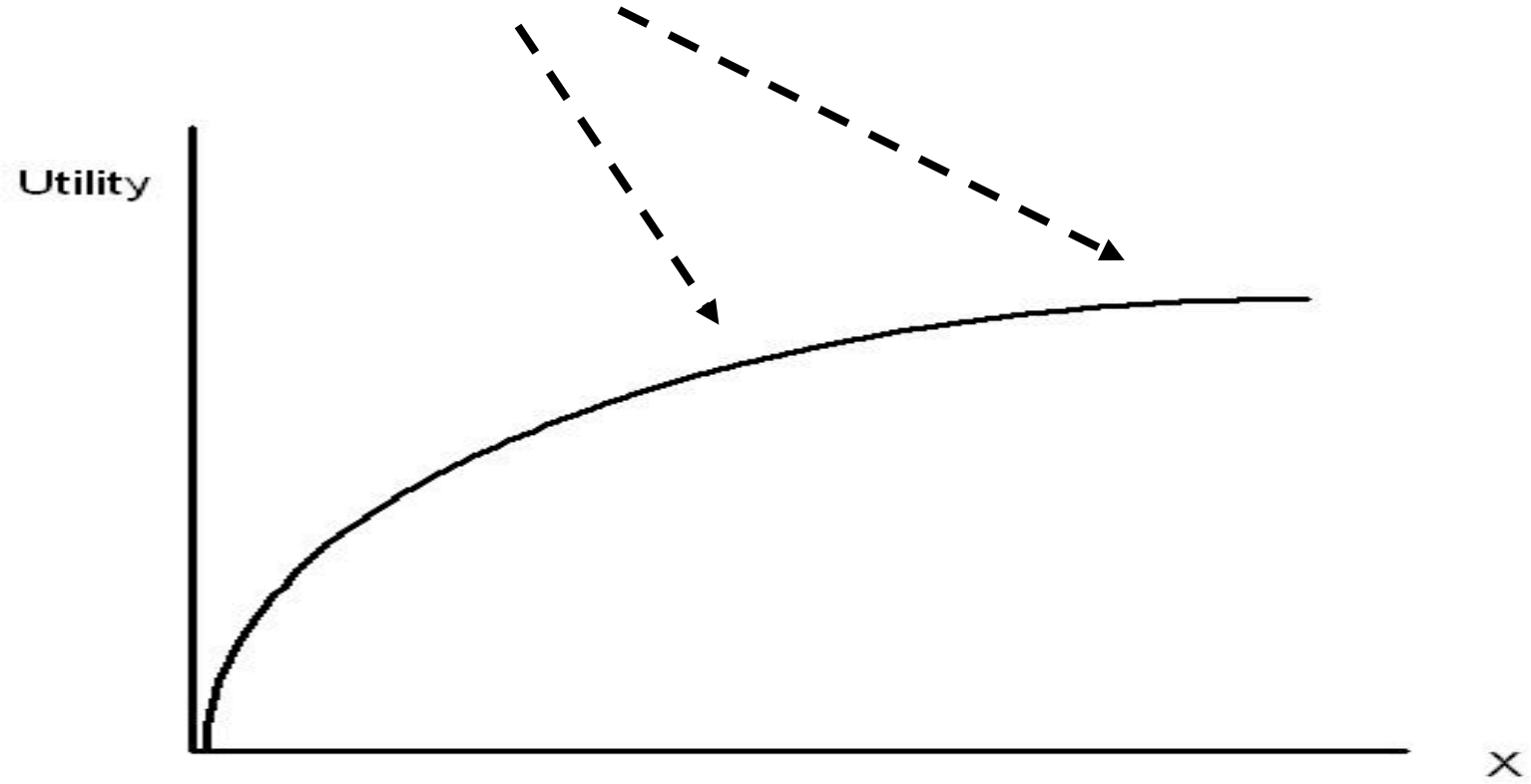
Start with a reasonable baseline model. Should be one with little effort but sophisticated enough to capture signals if they exist.



Model Iteration Cycle

ALWAYS BE AGILE: ITERATE

Iterate towards better models: in steps 2 – N, try new features and new algorithms. Always start with a good evaluation framework that is grounded in experimental design.



Model Iteration Cycle