Random Forests
What, Why, And How

Andy Liaw
Biometrics Research, Merck & Co., Inc.
andy_liaw@merck.com

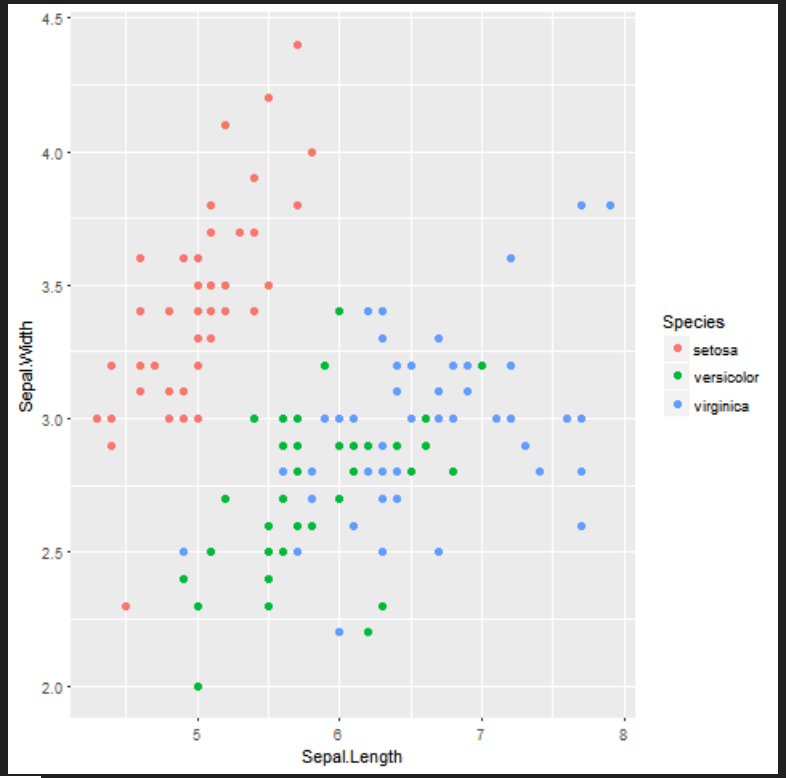
Outline

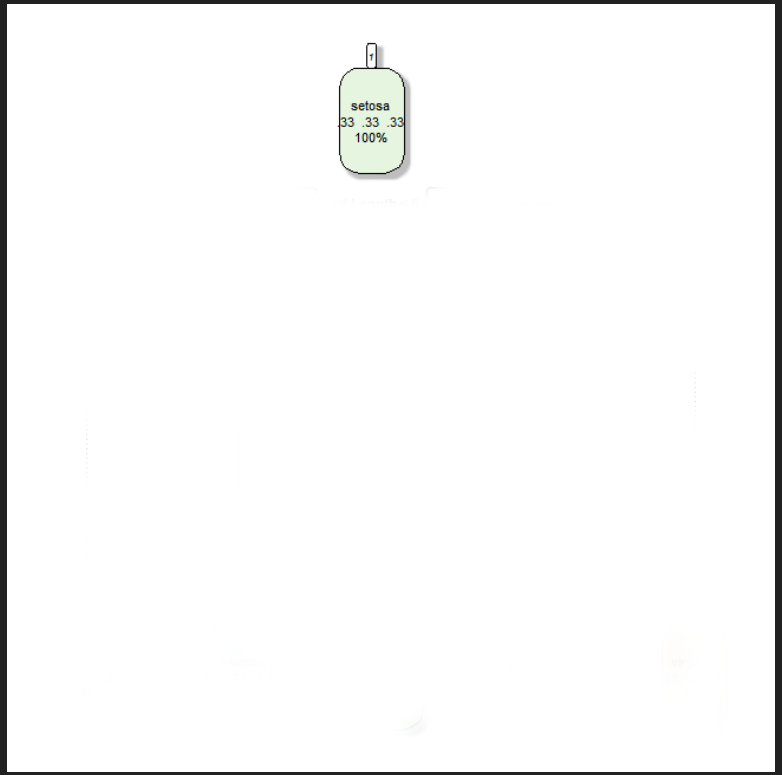
- Brief description of random forests
- Why does it work?
- Tuning random forests
- Comparison with other algorithms
- Gravy
- Wrap up

CART

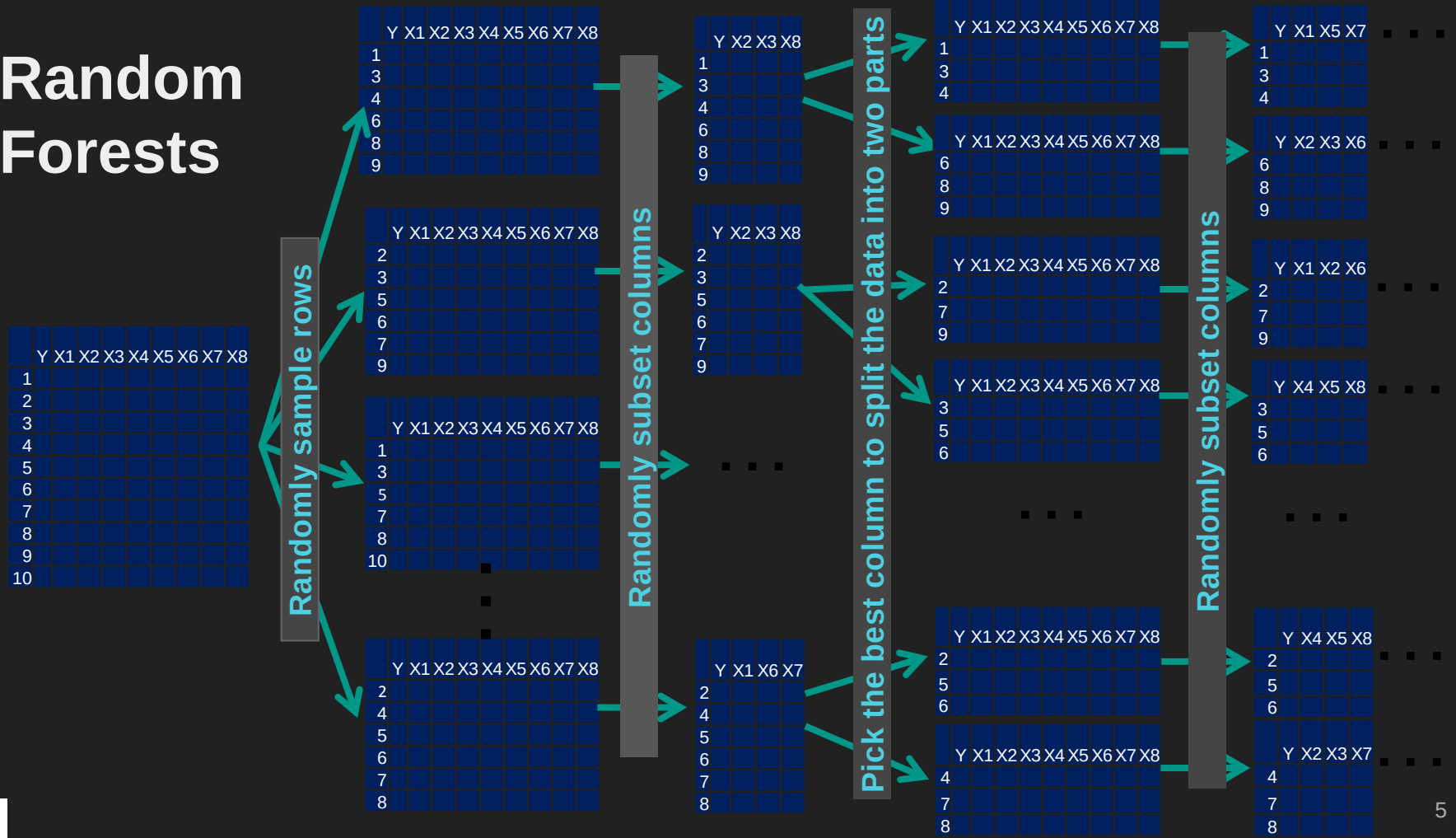
- Find best “gap” in a variable to split the data into two parts
- Repeat until futile
- Naturally handle categorical and numerical variables
- Very greedy algorithm => unstable
- Algorithm can be parallelized at different levels
- Finding “right-sized” tree requires cross-validation
- Generally not very accurate

Example: Iris Data





Random Forests



CART

Random Forest

Why Is RF Popular?

- Inherits many advantages of CART: places very little requirement on data preprocessing
- High Performance
- “Does not overfit”

Journal of Machine Learning Research 15 (2014) 3133-3181 Submitted 11/13; Revised 4/14; Published 10/14

Do we Need Hundreds of Classifiers to Solve Real World Classification Problems?

Manuel Fernández-Delgado MANUEL.FERNANDEZ.DELGADO@USC.ES
Eva Cernadas EVA.CERNADAS@USC.ES
Senén Barro SENEN.BARRO@USC.ES
CITIUS: Centro de Investigación en Tecnoloxías da Información da USC
University of Santiago de Compostela
Campus Vida, 15872, Santiago de Compostela, Spain

Dinani Amorim DINANIAMORIM@GMAIL.COM
Departamento de Tecnologia e Ciências Sociais- DTCS
Universidade do Estado da Bahia
Av. Edgard Chastinet S/N - São Geraldo - Juazeiro-BA, CEP: 48.305-680, Brasil

How Ensemble Models Work

Every model need to be better than random guessing

Try to have different model make mistakes on different data

Y	Model1	Model2	Model3	Model4	Model5	Aggregate
0	0	1	0	0	1	0
1	0	1	1	1	1	1
1	1	1	0	1	1	1
0	1	0	0	1	0	0
0	0	0	1	0	1	0
1	1	1	1	0	0	1
	67%	83%	67%	67%	50%	100%

Correlation and Strength

Correlation: How similar the base predictors are

Strength: How accurate the base predictors are

Low correlation => high diversity

Correlation comes with sufficiently high strength

Find good compromise between the two

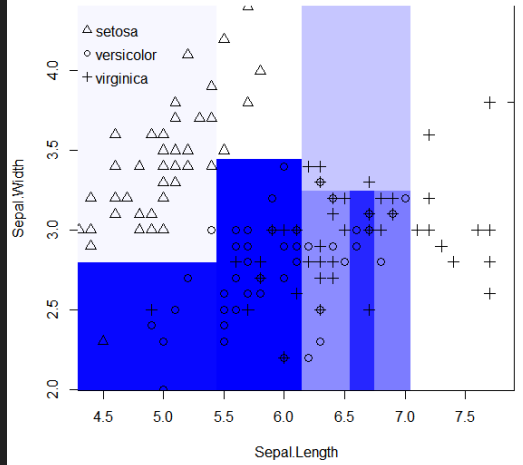
Small `mtry` promotes diversity

Nearest Neighbor Classifier

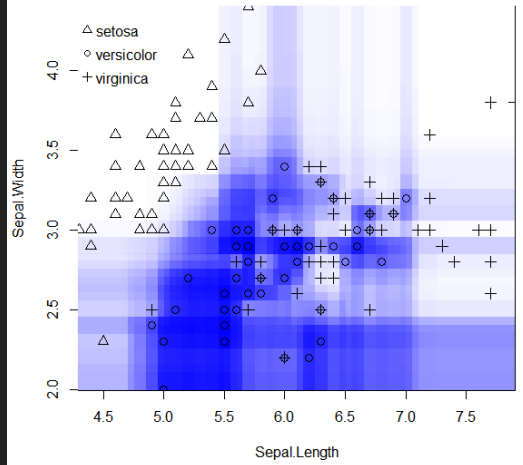
Terminal nodes in a decision tree represent groups of similar data, with sizes of neighborhoods decided from training data (hyper-rectangular regions)

RF averages terminal nodes from many trees, so neighborhoods are varied -- “smooth out” the crude neighborhoods of a single tree

CART



RF



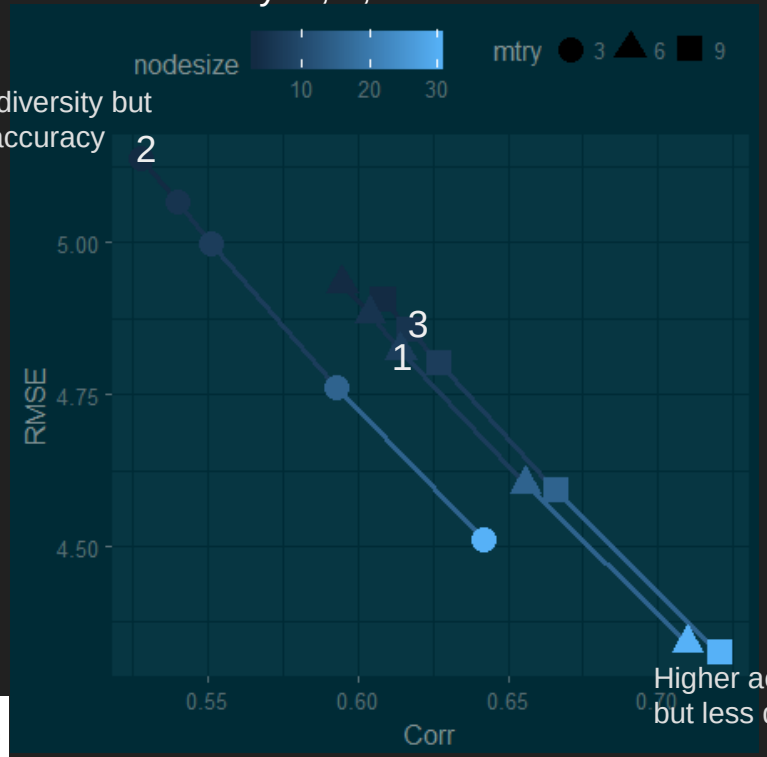
What Controls RF’s Model Complexity?

- Viewed as adaptive weighted NN, increasing number of trees makes weights “smoother”
- Sizes of neighborhoods can also be an indicator of model complexity
- There is evidence that smaller trees in RF work better for some data
- Given the same data, smaller trees ⇔ larger neighborhoods

Median Correlation vs. Median RMSE

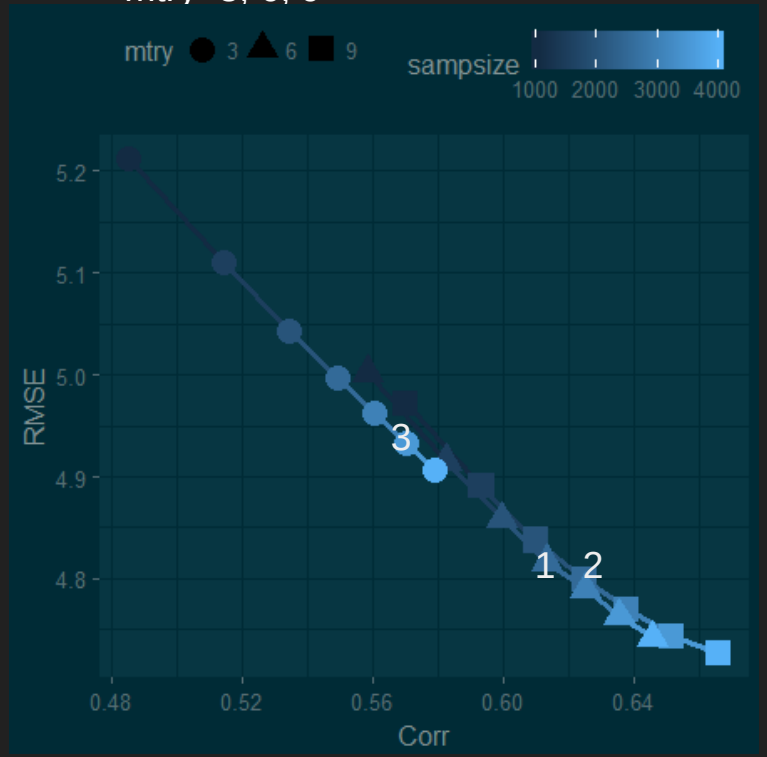
nodesize=3, 5, 7,15, 30
mtry=3, 6, 9

Higher diversity but worse accuracy



Higher accuracy but less diversity

sampsize=20%, 30%, …, 80%
mtry=3, 6, 9

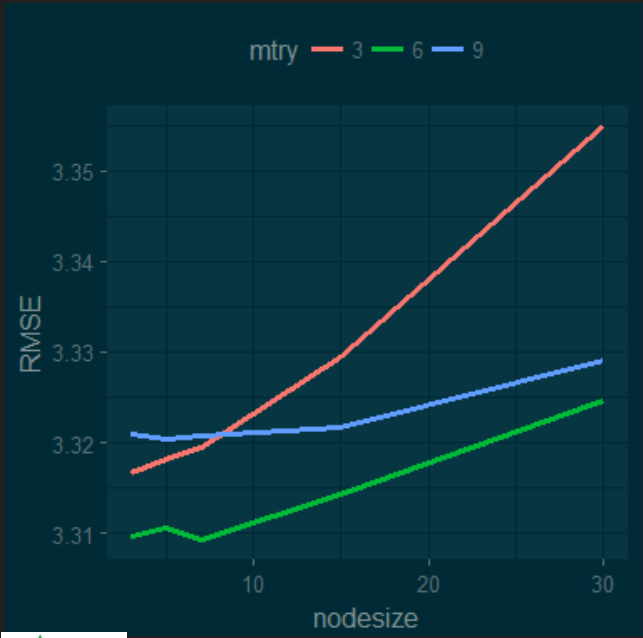


Tuning RF

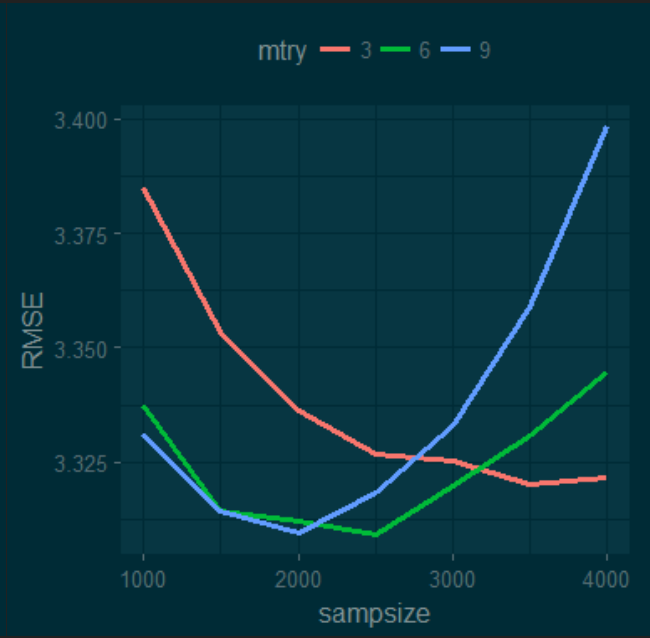
- Use `mtry` to balance correlation and strength
- A larger `nodesize` forces the algorithm to produce smaller trees, thus larger neighborhoods
- A smaller `sampsize` also induces smaller trees, also make trees more diverse (but should be used with larger number of trees)

Simulated Example: Friedman #1

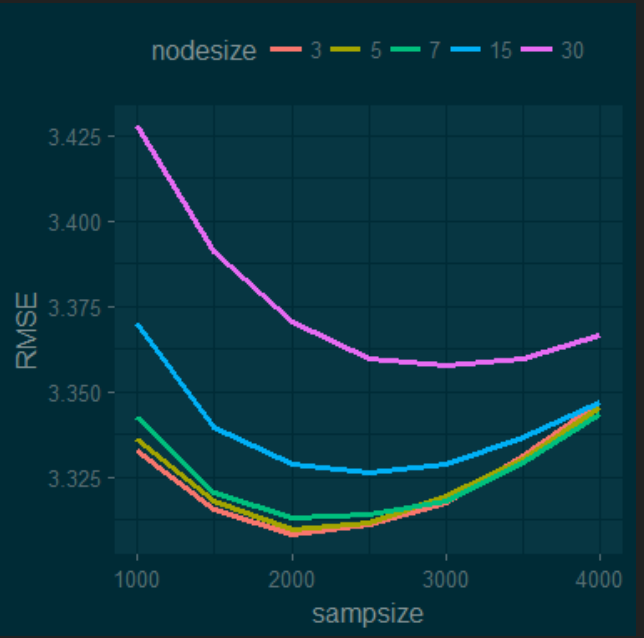
nodesize=3, 5, 7,15, 30
mtry=3, 6, 9



sampsize=20%, 30%, …, 80%
mtry=3, 6, 9



sampsize=20%, 30%, …, 80%
nodesize=3, 5, 7, 15
mtry=6



“Does Not Overfit”

- Definition of “overfitting”?
- Very different from something like Neural Networks
  - Early-stopping: monitor difference between training and validation errors; divergence of the two is seen as overfitting
- RF grows each tree to maximum size, thus have nearly 0 training error
- For boosting, test set error can keep decreasing as iterations go on even after training error reached 0! (But it will eventually increase– can’t boost forever)
- Bottom line: gap between training error and test set error does not necessarily indicate overfitting; increasing test set error with increasing model complexity does

RF vs. Boosting

RF

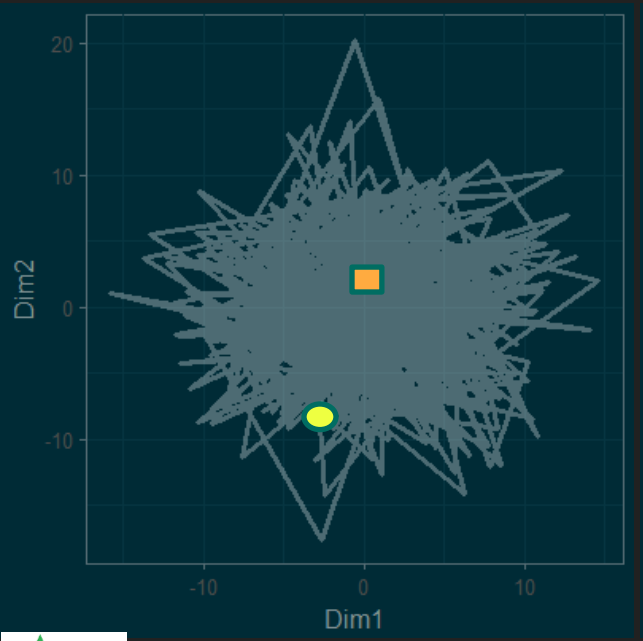
- Trees are independently grown
- Use randomness to get diverse trees
- Grow trees to maximum sizes
- Number of trees is not a tuning parameter
- Model size can be huge due to maximal size trees

Boosting

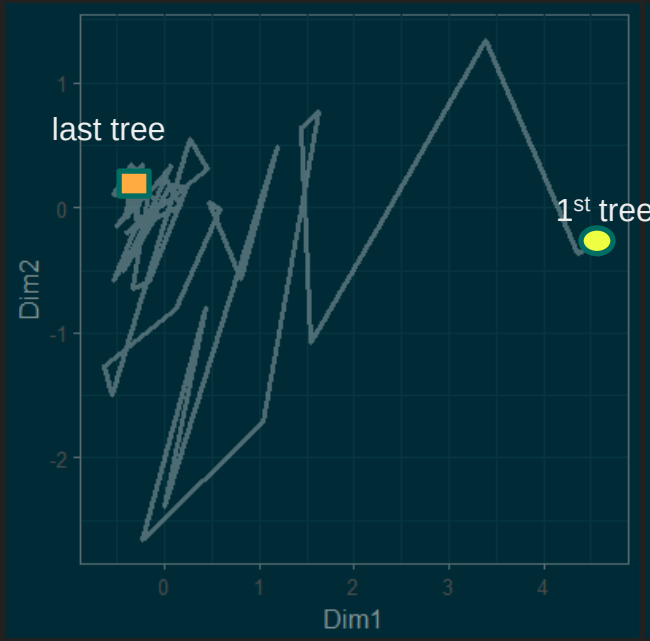
- Trees are grown sequentially
- Each tree tries to correct previous mistake
- Keep each tree relatively small
- Number of trees should be tuned
- Model size is usually small

MDS Projections of Individual Tree Predictions

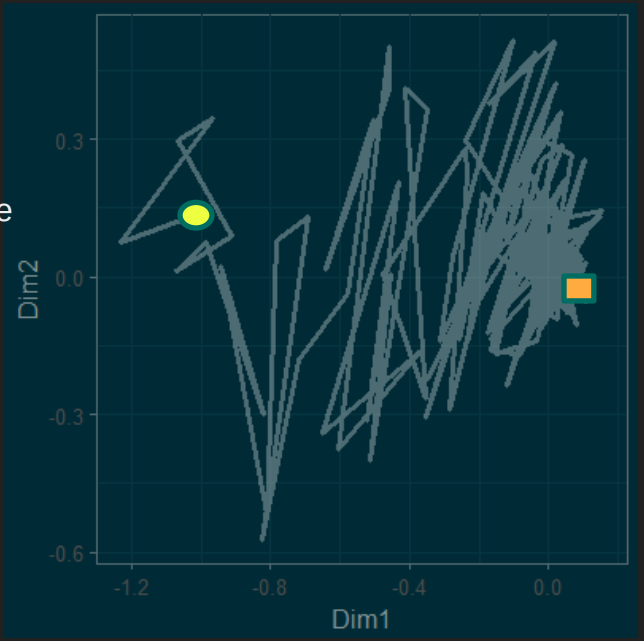
RF



GBM (100 trees, shrinkage=.2)



GBM (700 trees, shrinkage=.05)



RF vs. DNN

RF:

Adding more trees to RF does not seem to increase model complexity

No explicit “optimization”

Prediction can not exceed the range of training data

Difficult to update model with new data

DNN:

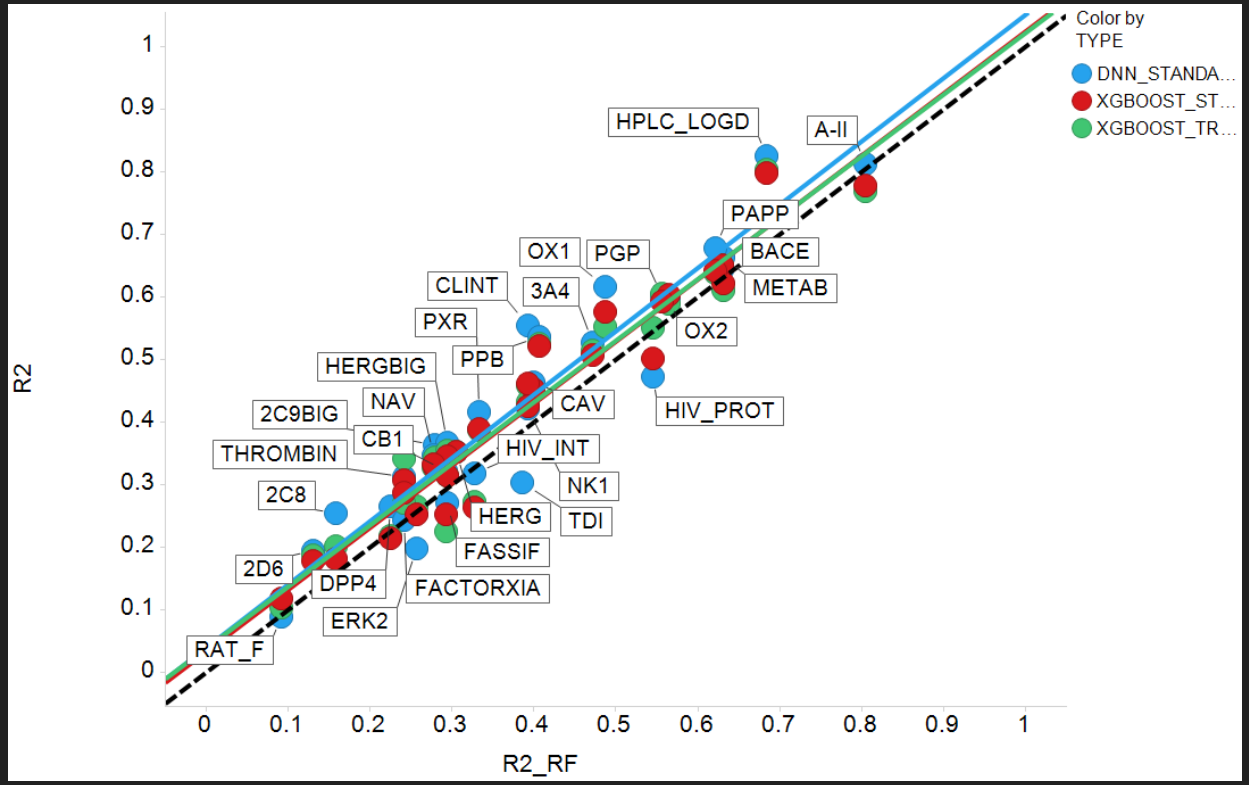
Complexity is predetermined by network architecture

Optimization with controlled greed

Prediction can be unbounded, depending on activation function

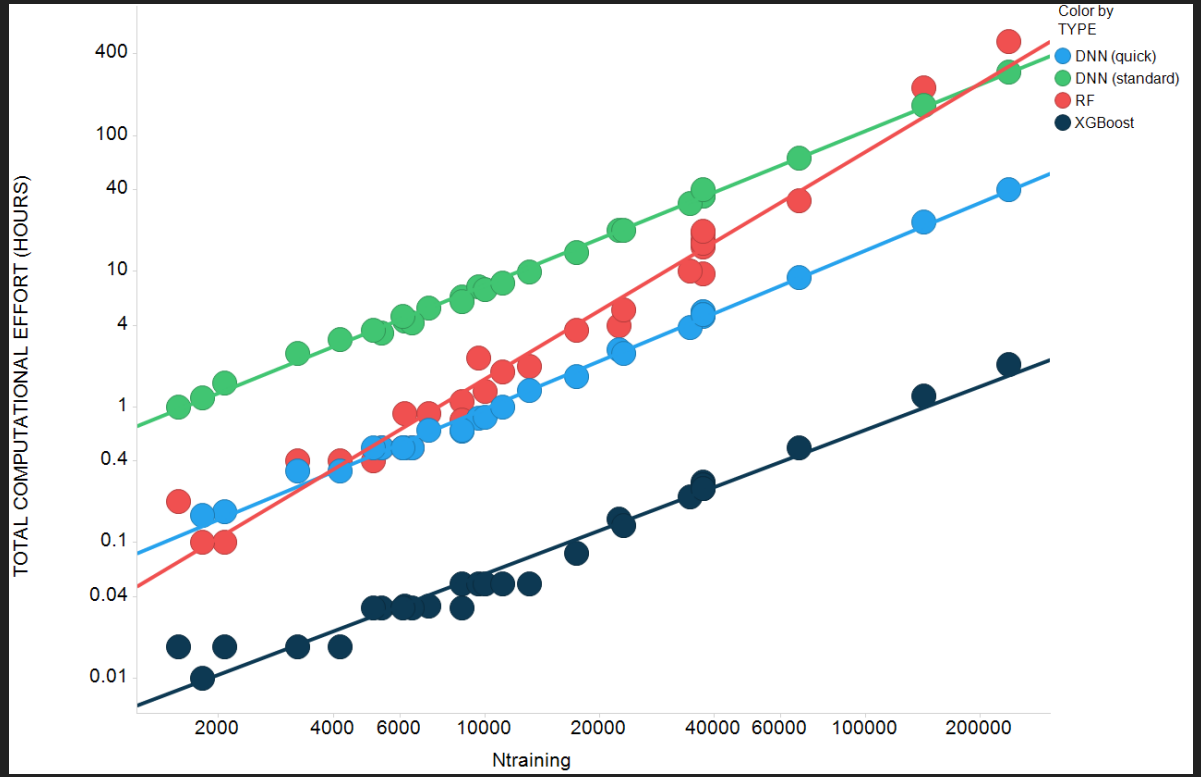
Trivial to update model with new data

RF vs. XGBoost vs. DNN: Performance



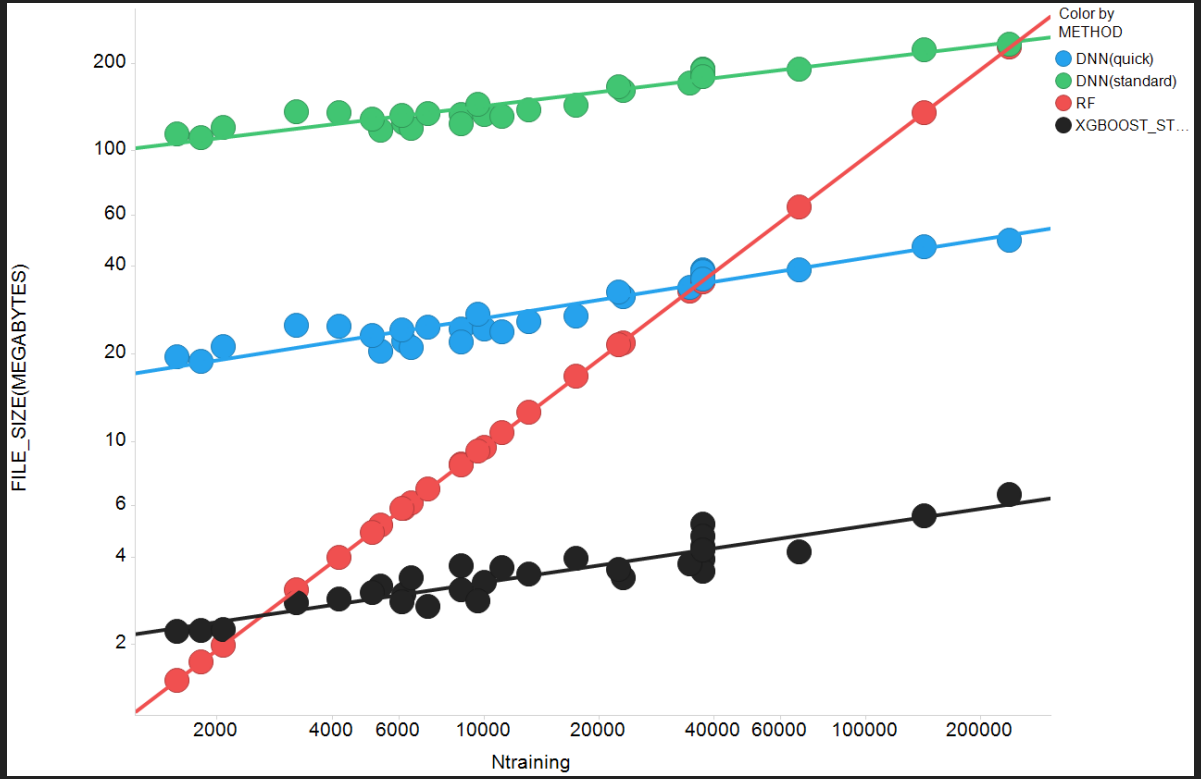
Courtesy of R. P. Sheridan

RF vs. XGBoost vs. DNN: Training Time



Courtesy of R. P. Sheridan

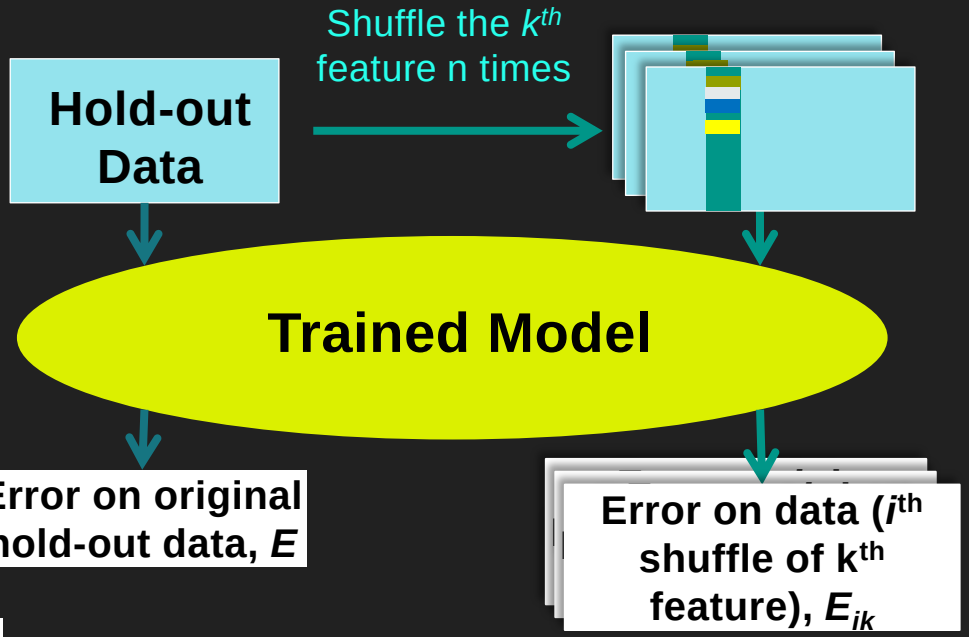
RF vs. XGBoost vs. DNN: Model Sizes



Courtesy of R. P. Sheridan

Variable Importance by Permutations

Breiman's idea of permuting data one variable at a time and seeing how accuracy drops can be apply to any algorithm, not just RF



For the k^{th} feature at the i^{th} shuffle:

$$d_{ik} = E_{ik} - E,$$

$$\text{VarImp}(F_k) = \frac{mean(d_{ik})}{SD(d_{ik})}$$

Partial Dependence Plot

- Every predictive model represents a function with multiple variables $y = f(x_1, \dots, x_p)$
- The marginal relation between y and a particular variable/predictor x_i can be examined using ***Partial Dependence*** (proposed by Friedman 1999).

Example:

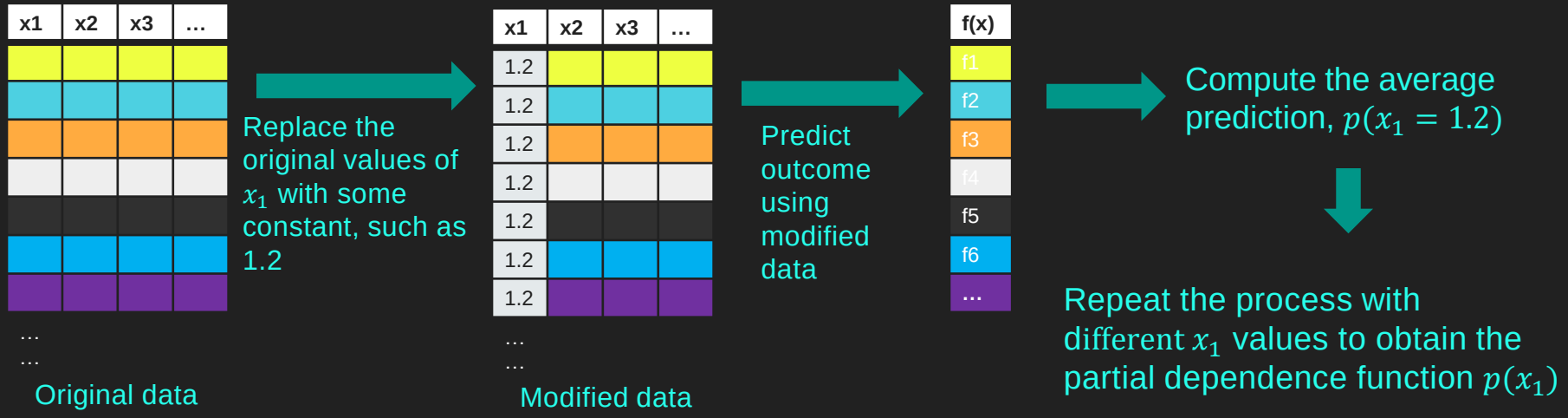
Assuming a model with 2 predicators, $y = f(x_1, x_2)$,

The **partial dependence** on x_1: $\boldsymbol{p(x_1)} = \int f(x_1, x_2)\, dx_2$

i.e., all remaining variables are integrated out

Computing Partial Dependence

Computing the partial dependence of variable x_1 for model $y = f(x_1, \ldots, x_p)$



Note: R package “pdp”, “ALEPlot”, and “ICEbox” implement this and extensions

Prediction Intervals

- The idea behind quantregForest enable prediction intervals to be formed by post-processing a randomForest object
    - For each new data point to be predicted, it lands in a leaf in each tree and is predicted by the mean of the (in-bag) data in that leaf, then averaged over all trees
    - We can use the in-bag data that fell in the same terminal nodes as the new data point as a sample from the conditional distribution, thus can be used to estimate the conditional quantiles
    - The **grf** package takes this idea further and use it for local likelihood
- While it might be possible to get such intervals with other methods by customizing the loss function to estimate quantiles, it requires fitting a separate model for each quantile

Room for Improvement

- Classification runs faster than regression, due to lack of pre-sorting in regression
- Currently tree depth is not tracked, thus cannot easily control it
- Splitting criteria are hard coded, no easy way to customize
- Handling of large number of categories is tricky
- Missing value handling can be better
- Some special tricks can speed up algorithm for some specific data type (e.g, all binary predictors)

Wrapping Up

- RF is a flexible, robust and high performance ML method
- Basic understanding of how it works can give intuitions on how to tune it
- For large data, try small **`sampsize`** and larger number of trees
- Some of the ideas introduced with the method can be extended to other methods

Acknowledgement

- Leo Breiman
- Adele Cutler
- Vladimir Svetnik
- Matt Wiener
- Numerous former interns
- Users who reported bugs