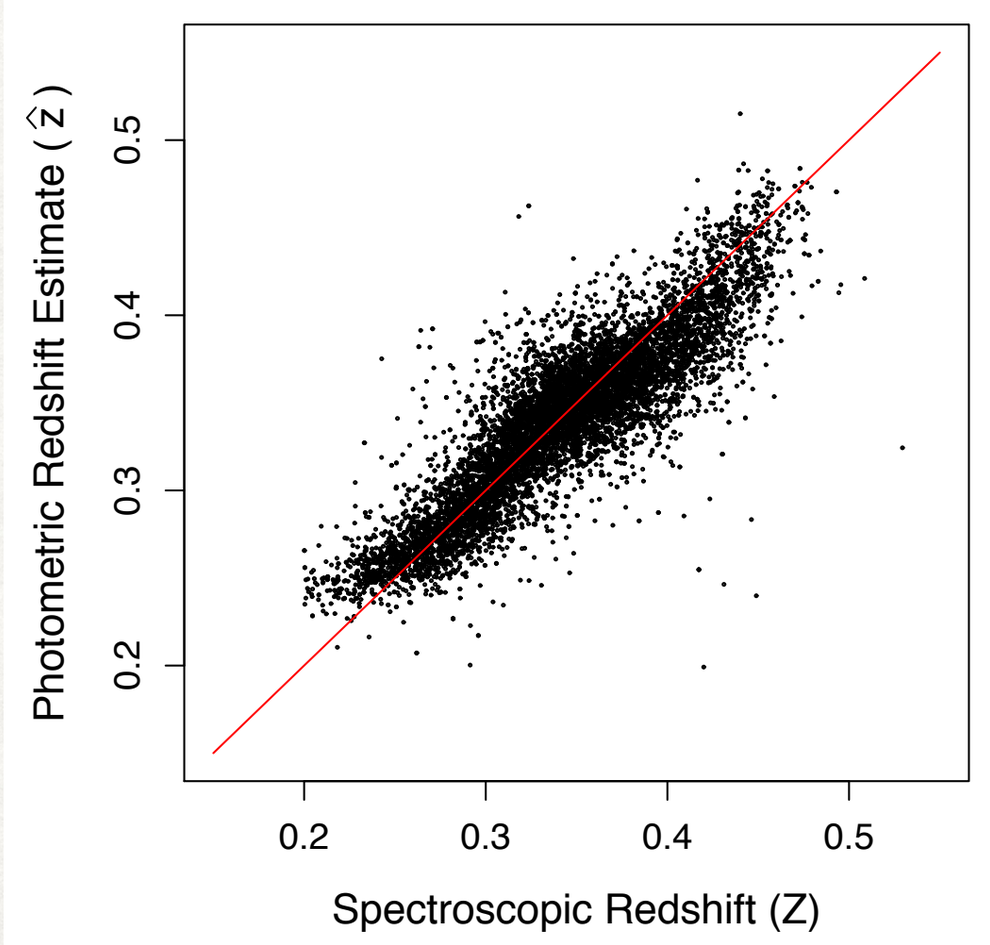
Machine Learning with Random Forest

Peter E. Freeman
Department of Statistics & Data Science
Carnegie Mellon University

From

```
u g r i z y redshift
17.8313 16.9077 16.4431 16.2099 16.0613 15.8732 0.038356
19.0731 17.7448 16.9789 16.5288 16.2551 15.9531 0.058309
21.638 21.0106 20.8286 20.6283 20.6552 20.528 0.063701
20.5474 19.5542 19.2387 19.0568 19.0887 18.9865 0.059006
21.2378 20.6876 20.5661 20.4371 20.4799 20.4503 0.063202
22.4627 21.4597 21.0484 20.8274 20.7639 20.6385 0.057773
23.8221 22.895 22.5779 22.3543 22.3225 22.2038 0.061548
23.0491 22.1536 21.8791 21.6889 21.7044 21.6381 0.063769
23.6742 23.0346 22.7857 22.6116 22.5813 22.5462 0.061427
23.5684 22.6635 22.3825 22.1811 22.1842 22.1003 0.066216
22.8421 21.9752 21.6568 21.4507 21.3821 21.2972 0.062465
22.27 20.98 20.3657 20.0249 19.8772 19.665 0.059334
22.9572 22.1043 21.8311 21.6471 21.6476 21.6021 0.064125
22.6374 21.7366 21.3624 21.0996 21.0509 20.8953 0.062879
```

To



Let's Start With…

What is Statistical Learning?

Statistical learning is the process of ascertaining associations between groups of variables, most often between a group dubbed *predictor* (or independent) variables and a single so-called *response* (or dependent) variable. (This is an example of *supervised* learning.)

Examples of statistical learning algorithms include:

- Linear regression and its variants (e.g., lasso)
- Trees and its variants (e.g., random forest)
- Extreme gradient boosting
- Deep learning

What is the Goal of Statistical Learning?

This can be boiled down to…

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

If you care about the details of $\hat{f}(X)$, then your goal is statistical inference.

If you treat $\hat{f}(X)$ as a black box, then your goal is prediction.

And if your goal is prediction, you will want to explore machine learning.

So, then…

What is Machine Learning?

Short Version:

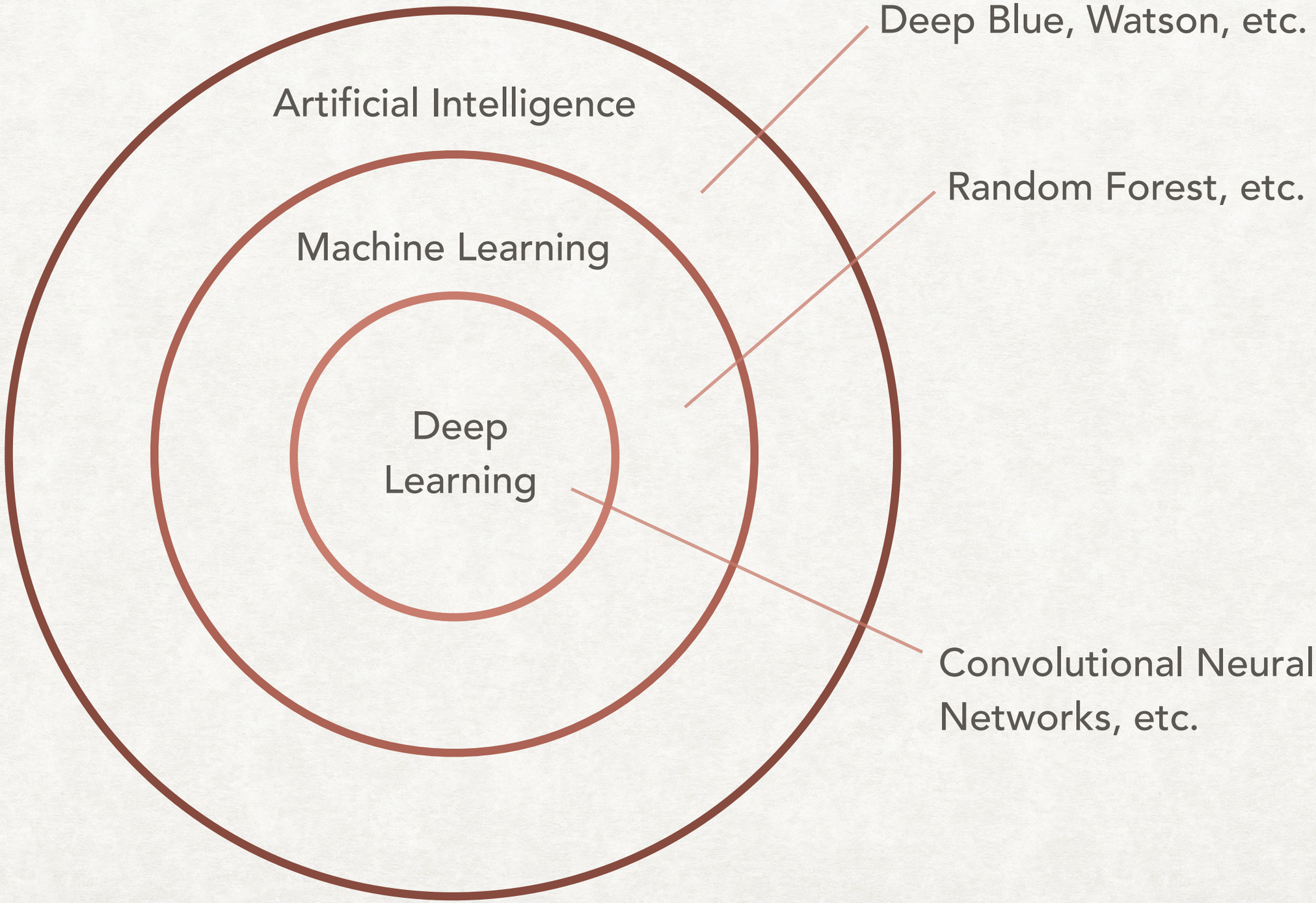
It is a subset of statistical learning that focuses on prediction.

Longer Version:

It is the idea of constructing *data-driven* algorithms that learn the mapping between the predictors and the response. Specifically, we suppose no parametric form for the mapping *a priori*, even if technically one can write one down *a posteriori*.

Thus linear regression is not machine learning (since the form of the mapping is fixed *a priori*), but random forest is ML (since we've no idea the number of tree splits that the algorithm will make while learning the relationship between predictors and response).

Broader Context: What is Machine Learning?



Machine Learning: Things to Keep in Mind

1. Different machine learning algorithms better adapt to different "data geometries." Unless you can visualize the full predictor variable space, it is best to try several algorithms in analyses.
2. Machine learning algorithms are not always better! If the relationship between predictors and response is linear, then a linear model will be the best one to learn.
3. You will hear the term "algorithmic bias" in relation to ML algorithms. This is a misnomer. An algorithm produces predictions given the data that are input into it, and biased predictions come from biased data. The term to remember is "selection bias": garbage in, garbage out.
4. ML algorithms, especially deep learning algorithms, require very large samples for the learning to be effective. Don't just default to deep learning…you may not have sufficient data. (And see #2.)

Simple Machine Learning: A Decision Tree

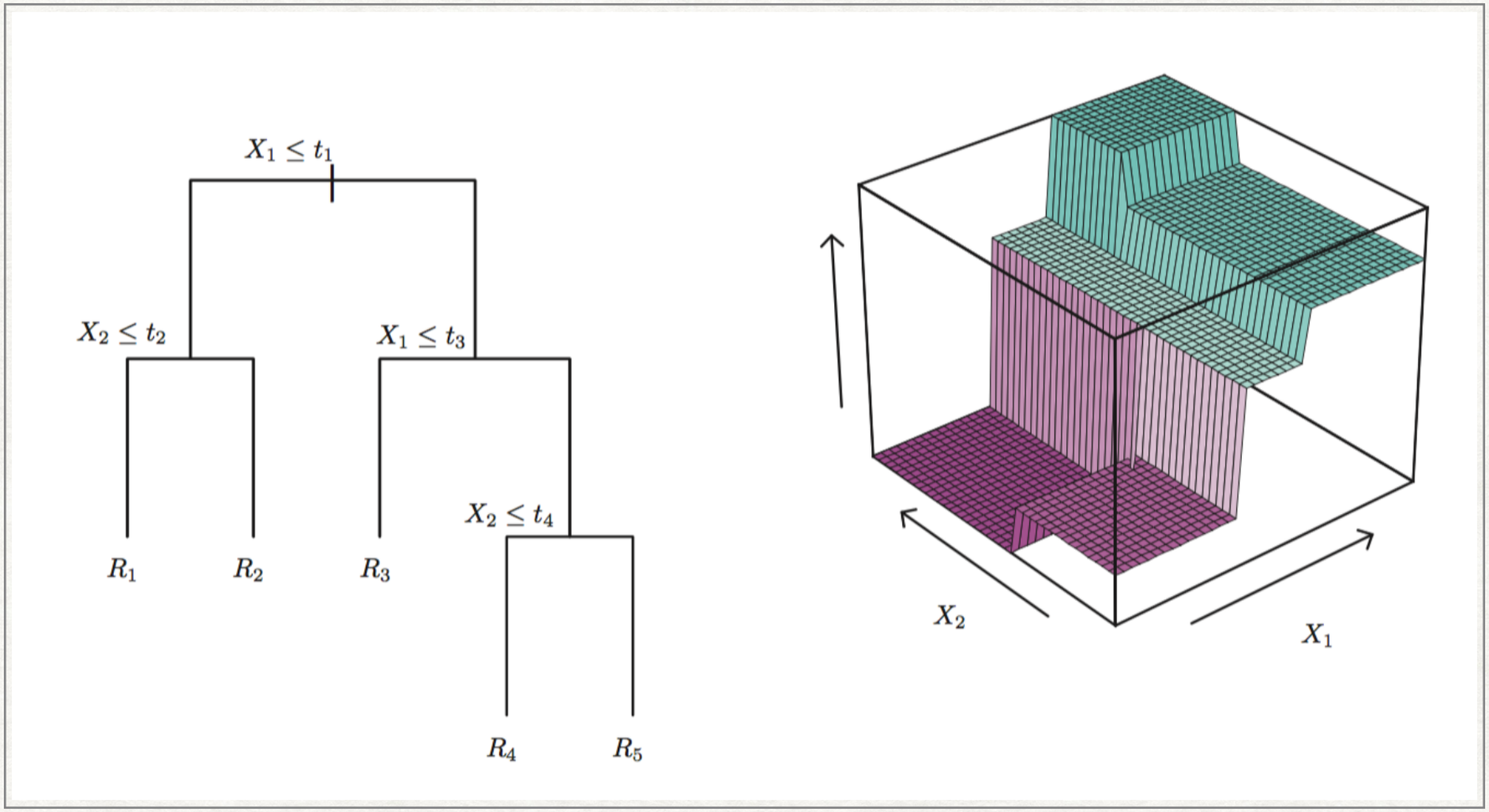


Figure 8.3, Introduction to Statistical Learning (James et al.)

The Decision (Regression) Tree Algorithm

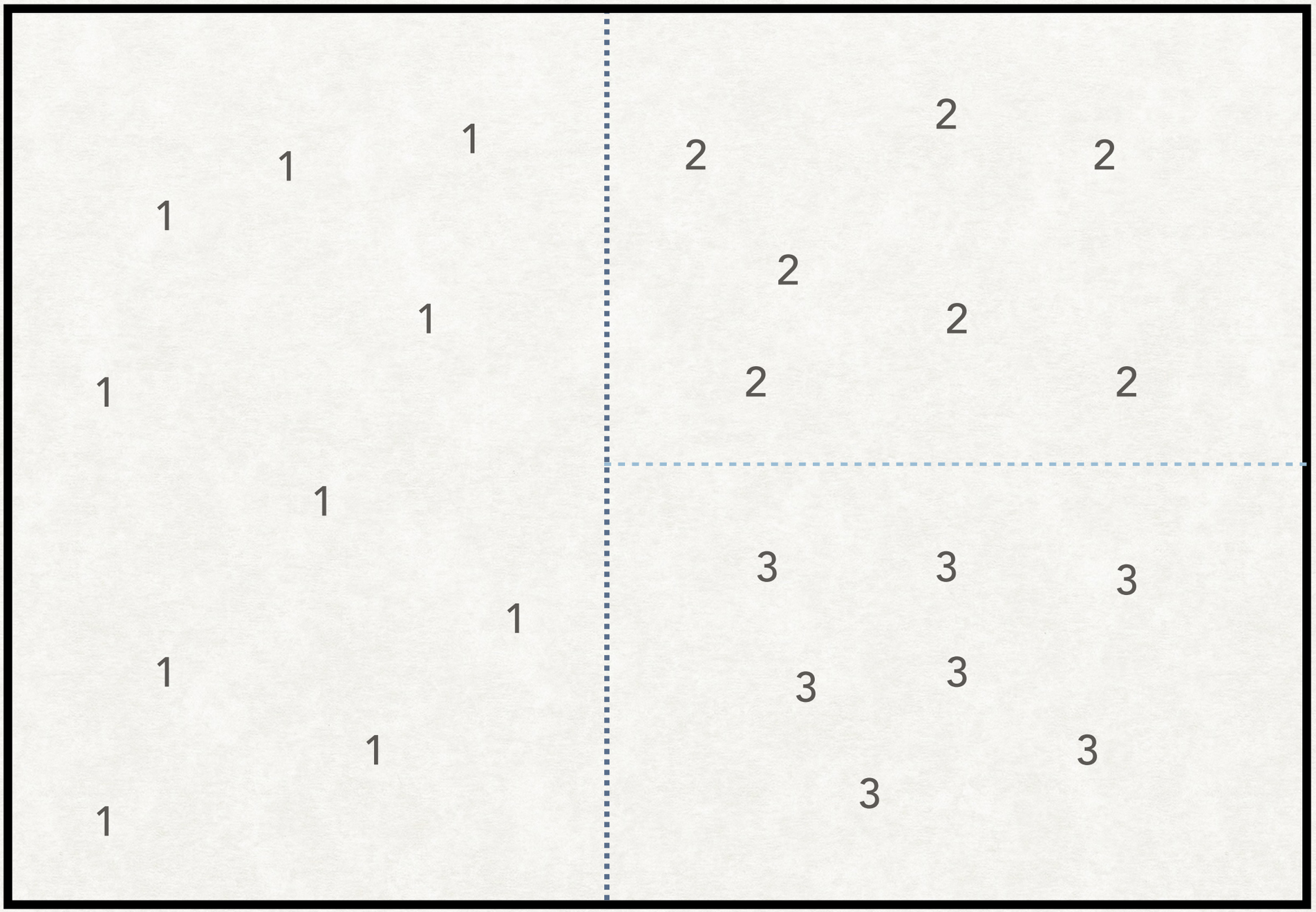
1. Compute the residual sum of squares (RSS) assuming that the best model is the mean of the response variable values.
2. Move along each axis in the space of predictor variables and split the data into two groups at each point.
3. For each split, recompute the means, and RSS, for each group. Sum the RSS values and set aside.
4. Determine which split affected the greatest reduction in RSS overall. If the reduction is sufficient, permanently split the data into two groups and for each group, repeat steps 1-3. If not, stop.

If instead our response variable has, e.g., *C* classes, we would not use RSS, but rather the Gini index:

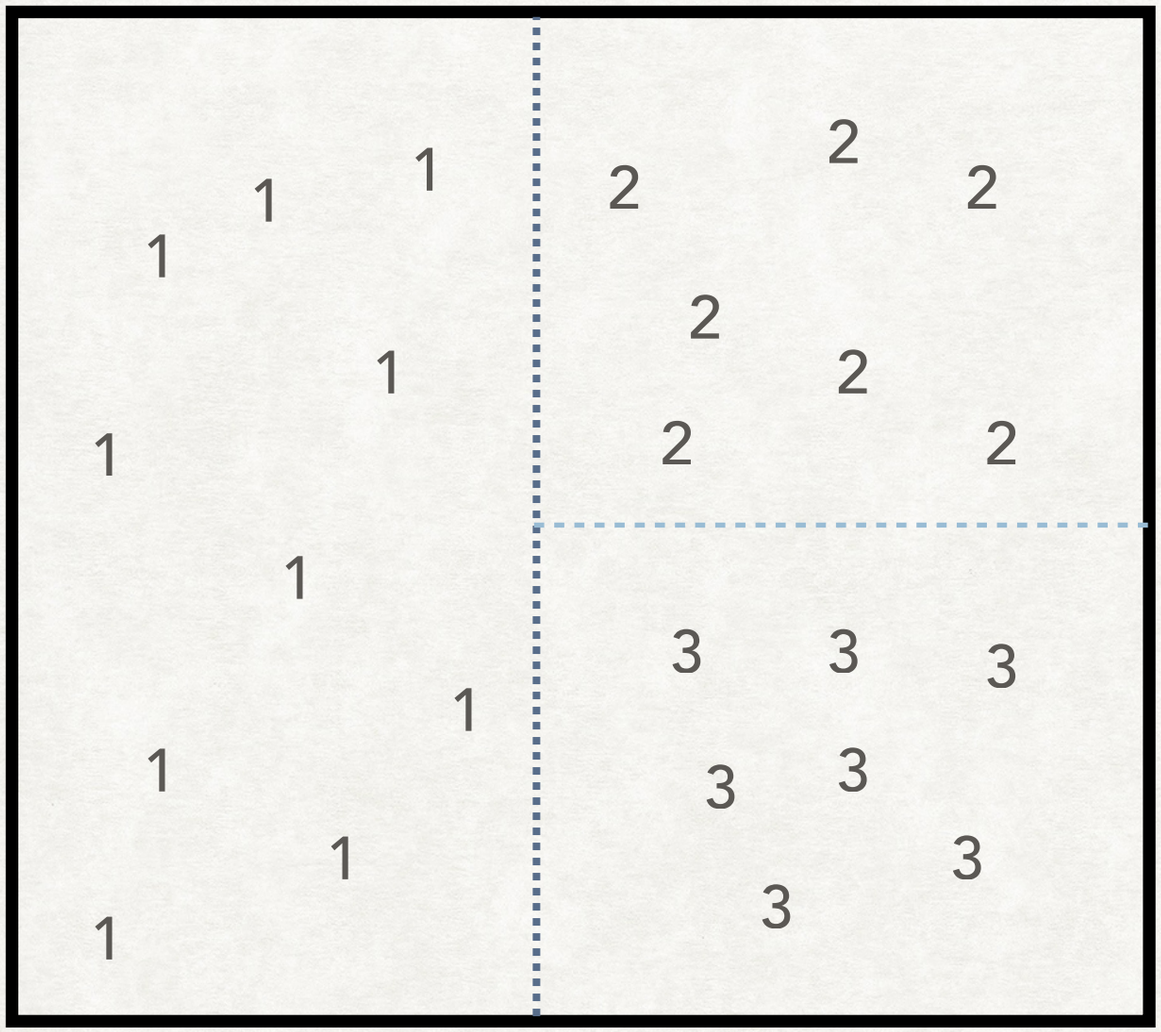
$$G = \sum_{c=1}^{C} \hat{p}_c(1 - \hat{p}_c)$$

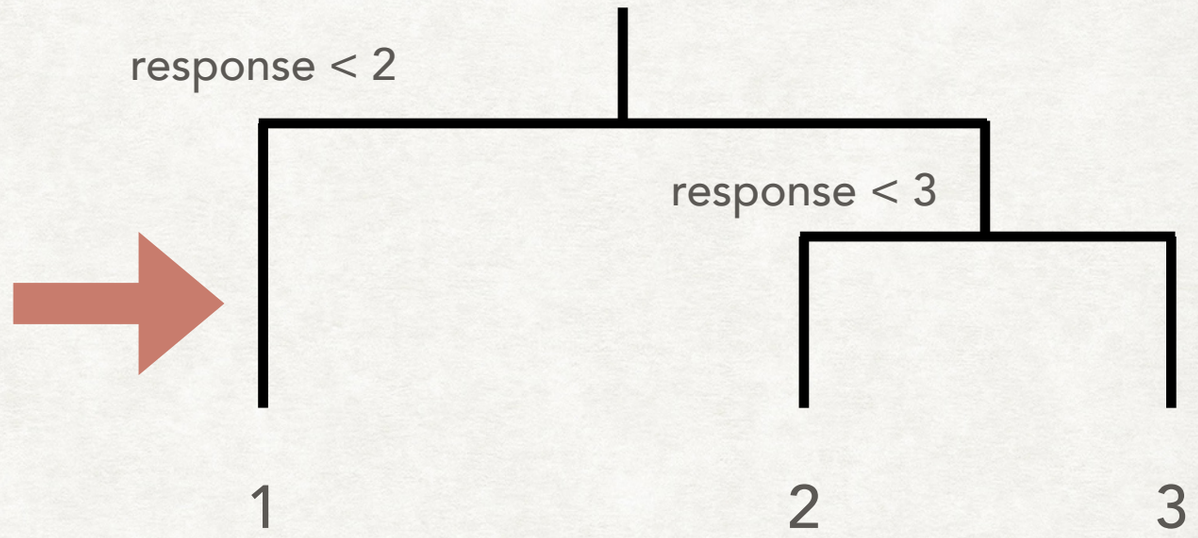
Here, p_c is the proportion of (training set) observations of class *c* in the node.

The Decision (Regression) Tree Algorithm



The Decision (Regression) Tree Algorithm





Did I Say Training Set?

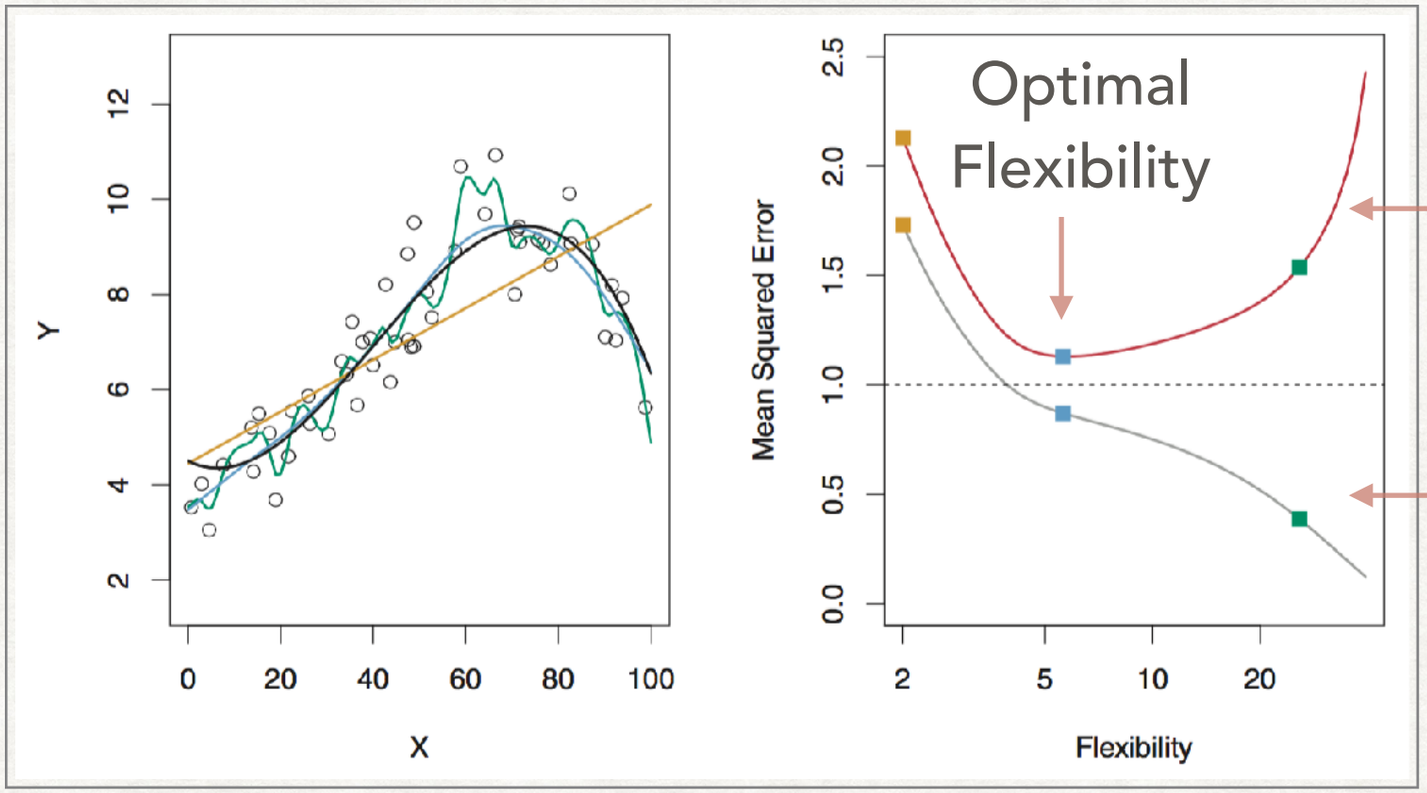


Figure 2.9, Introduction to Statistical Learning (James et al.)

An arbitrarily flexible statistical learning model will overfit data. Ergo, one cannot apply a tree, for instance, to the entire dataset; rather, one holds data out (in a so-called test set) and assesses model performance using those data.

Did I Say Training Set?

The two most oft-used training "paradigms":

1. Split the data into training and test datasets at the beginning. There is no overriding convention, but a 70-30 split is typical.
2. Split the data into *k* groups (or folds) in the beginning, then loop over the folds: hold the i^{th} fold out as a test set, then loop over the others. This is *k*-fold cross validation. The optimal number of splits is 5-10.

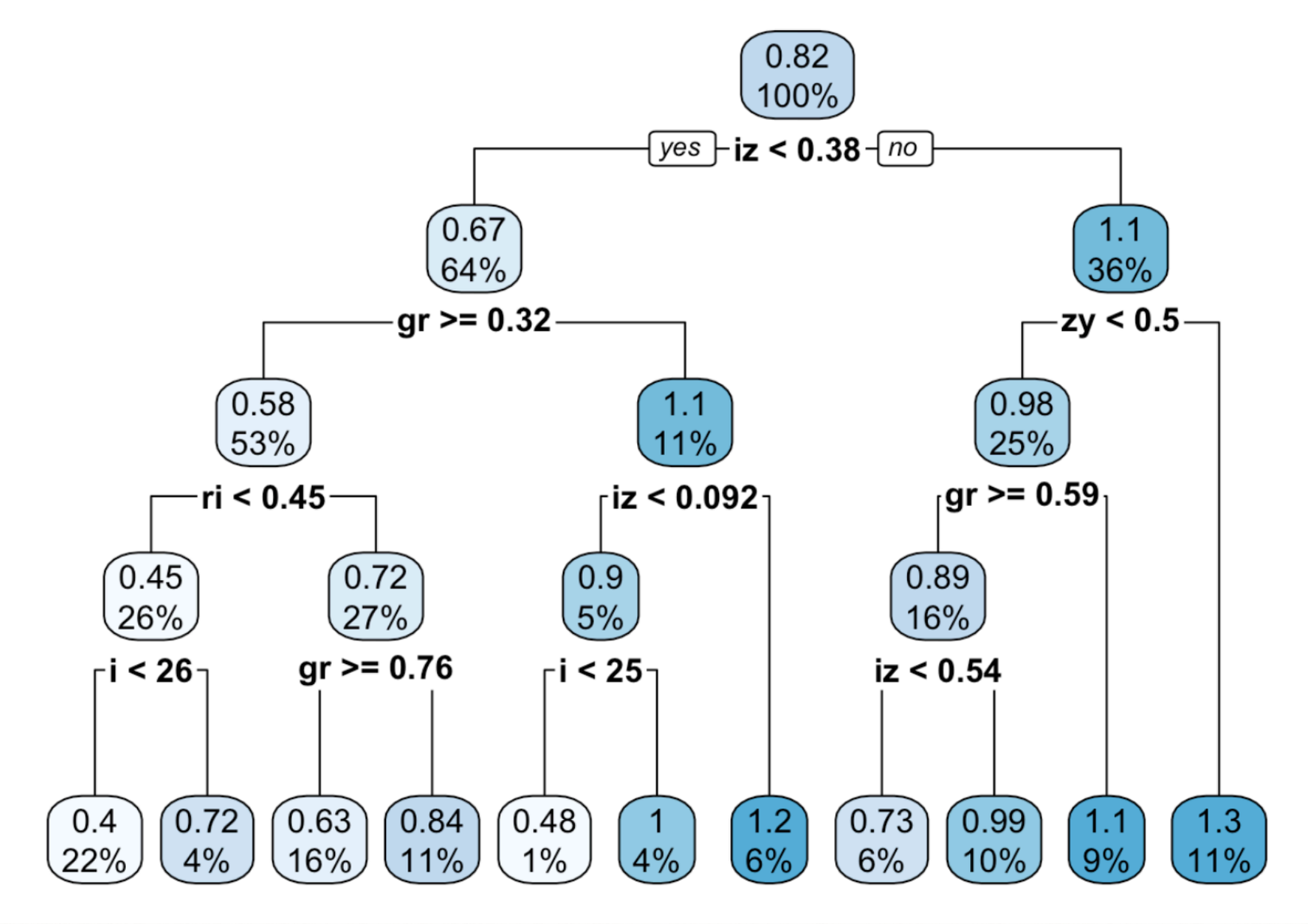
k-fold CV is more computationally intensive but produces model assessment metrics (like mean-squared error) with less variance.

NOTE: for reproducibility, set a random number seed ***before*** assigning splits or folds! And if comparing models, use the ***same*** splits or folds throughout model selection!

Lab Exercise: Decision Trees

- Dataset: Rubin Observatory Data Challenge 1 (DC 1)
- 9065 (simulated) galaxies
- Six predictor variables:
  - colors: *u-g*, *g-r*, *r-i*, *i-z*, *z-y*
  - magnitude: *i*
- Two possible response variables: redshift and stellar mass
- 70-30 training-test split

Lab Exercise: Decision Trees



Test-set MSE: 0.088 (or uncertainty in redshift is ~0.3)

Decision Trees: Why and Why Not

Why:

Easy to visualize and interpret!

Why Not:

They do not generalize well; e.g., the test-set MSE tends to be high.

Care must be taken to "prune" a tree to avoid overfitting test-set data.

Tree models are sensitive to dominant predictors: if one predictor has much but not all the explanatory power, the first split will always be made along that predictor's axis…potentially obscuring some of the explanatory power of the other predictors. (This is a pitfall of so-called *greedy algorithms* that make "locally" optimal choices that can lead to "globally" sub-optimal solutions.)

Random Forest

Short Version:

RF aggregates many trees (hence, a forest) into a single solution.

The Details:

"Wait…many trees? I only have one set of training data…"

IMPORTANT!

RF employs two tweaks:

1. New training data are created for each tree via bootstrapping, i.e., resampling the observed data with replacement.
2. For any single tree, only a subset of predictor variables are considered. (This is referred to as "decorrelating" the trees.)

NOTE! As we will see, RF tabulates "variable importance," which makes it an ML algorithm that allows some level of statistical inference.

Random Forest

In standard usage, there are no parameters to tune! RF is basically “plug ’n’ play.”

By default, 500 trees. Generally, any value above ~100 is fine.

The number of predictors for each tree: square root of the overall number.

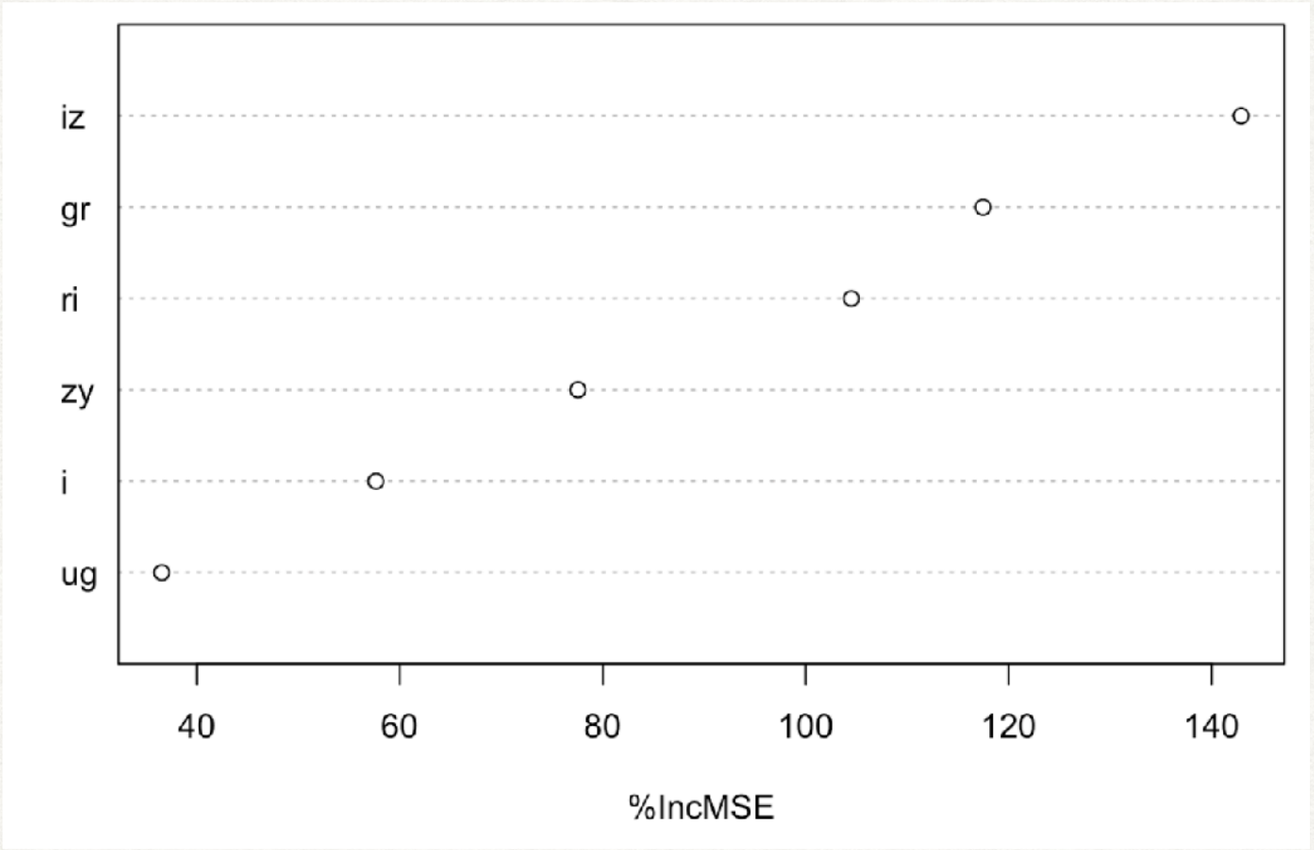
```
randomForest(x, y=NULL,  xtest=NULL, ytest=NULL, ntree=500,
             mtry=if (!is.null(y) && !is.factor(y))
             max(floor(ncol(x)/3), 1) else floor(sqrt(ncol(x))),
             replace=TRUE, classwt=NULL, cutoff, strata,
             sampsize = if (replace) nrow(x) else ceiling(.632*nrow(x)),
             nodesize = if (!is.null(y) && !is.factor(y)) 5 else 1,
             maxnodes = NULL,
             importance=FALSE, localImp=FALSE, nPerm=1,
             proximity, oob.prox=proximity,
             norm.votes=TRUE, do.trace=FALSE,
             keep.forest=!is.null(y) && is.null(xtest), corr.bias=FALSE,
             keep.inbag=FALSE, ...)
```

Set to TRUE to get importance measured via mean-squared error (MSE) for regression. (Even if FALSE, will get Gini-based importance for classification. Go figure.)

Approximately 63% of the training data are sampled for any one tree.

Each RF tree is grown “deep” and is not pruned. Aggregation “averages out” the effects of overfitting.

Lab Exercise: Random Forest



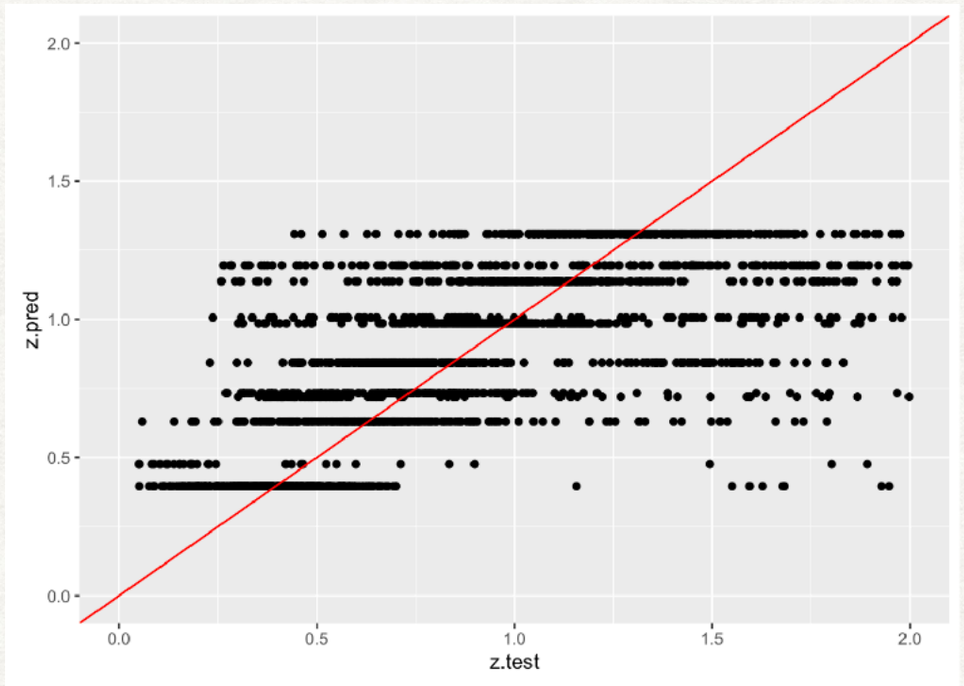
Variable importance plot

How much does the test-set MSE increase if we randomly permute the data in a given column? The more it increases, the more statistical information that variable contains.

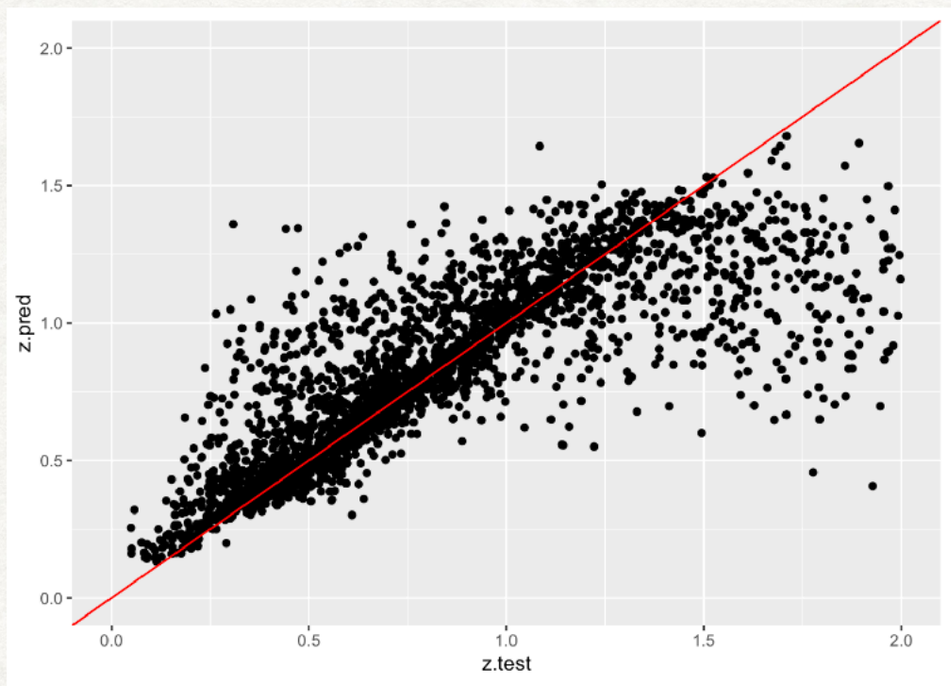
Test-set MSE: 0.061 (or uncertainty in redshift is ~0.25)

Lab Exercise: Decision Tree vs. Random Forest

Decision Tree



Random Forest



Diagnostic plot: predicted values versus observed values for test-set data

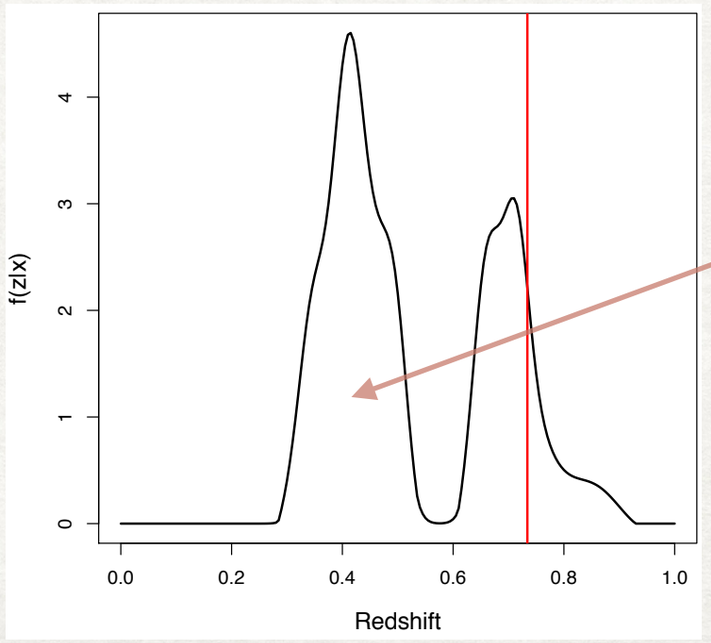
Regression vs. Conditional Density Estimation

For simplicity, let's assume for a moment that we have one predictor variable.

For a given value for that variable, e.g., x_o, there is a distribution governing the possible observed response values. (In simple linear regression, that distribution is assumed to be normal.) This is a conditional probability density function (or just a conditional density), and we can denote is $f(y \mid x_o)$.

In regression, the goal is to estimate the mean (or expected value) of $f(y \mid x_o)$.

Summarizing $f(y \mid x_o)$ using the mean is OK if $f(y \mid x_o)$ is indeed normal. But what if it is skew? multi-modal? just simply not normal? In that case, it can be useful to estimate $f(y \mid x_o)$ directly. This is conditional density estimation, or CDE.



Note: your CDE is working well if the areas to the left of the red lines (the true values) are distributed uniformly for your test objects.

Example of a conditional density estimate of a photometric redshift.

RFCDE: Random Forest-based Conditional Density Estimation

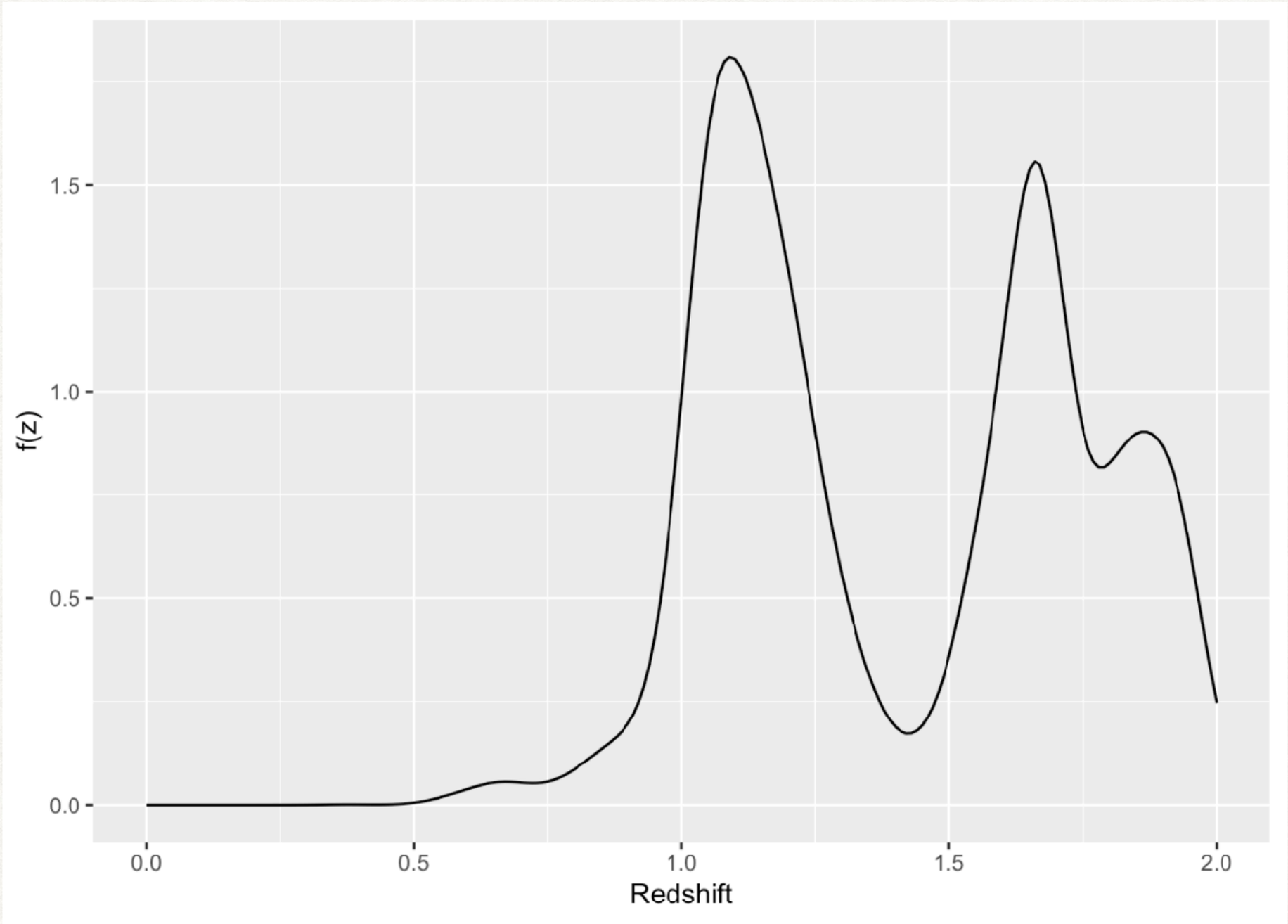
Pospisil & Lee (2018) - arXiv:1804.05753

https://github.com/tpospisi/RFCDE

Key Takeaway Points:

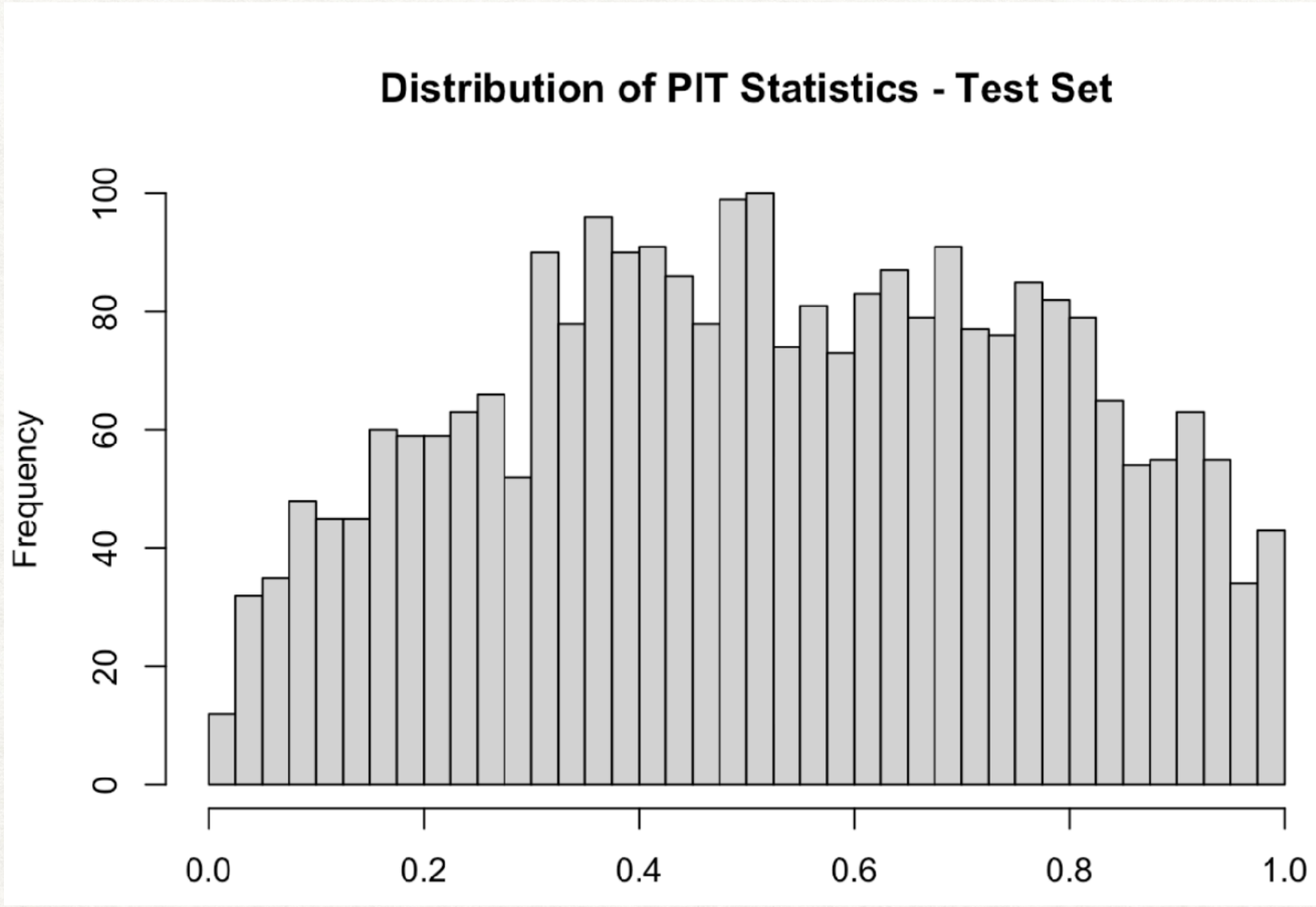
1. Rather than basing splits on a regression-based loss function (MSE), RFCDE utilizes a conditional density-based loss function.
2. RFCDE can be extended to multivariate settings, i.e., more than one response variable.

Lab Exercise: RFCDE



Example of conditional density function: $\hat{f}(z \mid u\text{-}g, g\text{-}r, r\text{-}i, i\text{-}z, z\text{-}y, i)$

Lab Exercise: RFCDE



Diagnostic: the shape of this distribution implies that the predicted CDE curves are systematically too wide: you see redshifts at lower and upper ends less than you would expect.

Summary

linear regression → decision trees → random forest

inflexible

inferential

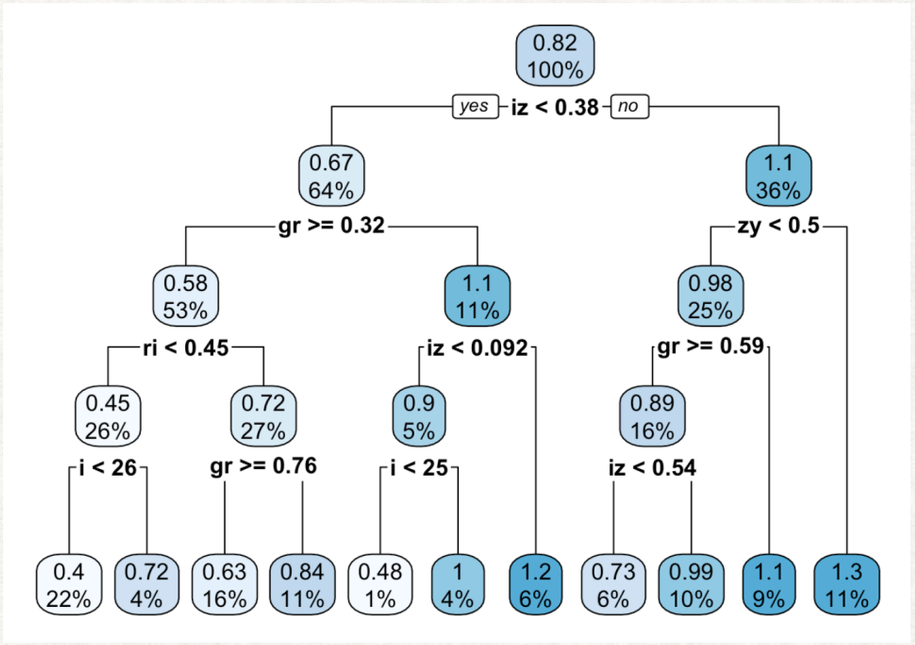
more flexible

better predictions (usually!)

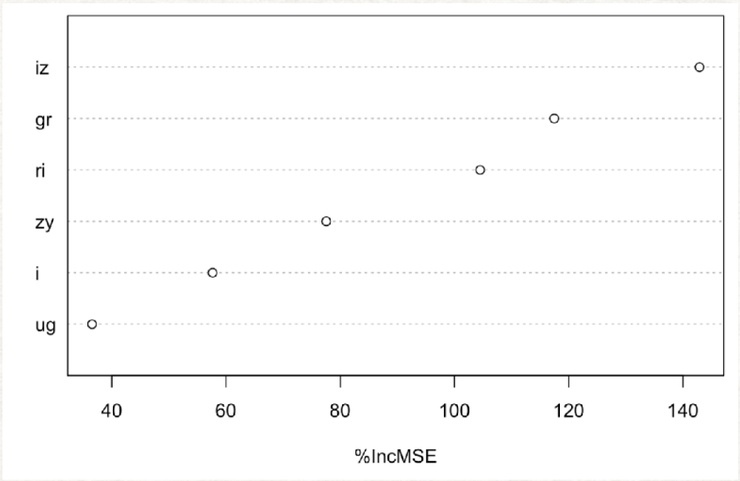
but only quasi-inferential

more easily visualizable output

but lesser predictive ability



vs.



variable importance