Random Forests™

Terence Parr
MSDS program
University of San Francisco

UNIVERSITY OF SAN FRANCISCO

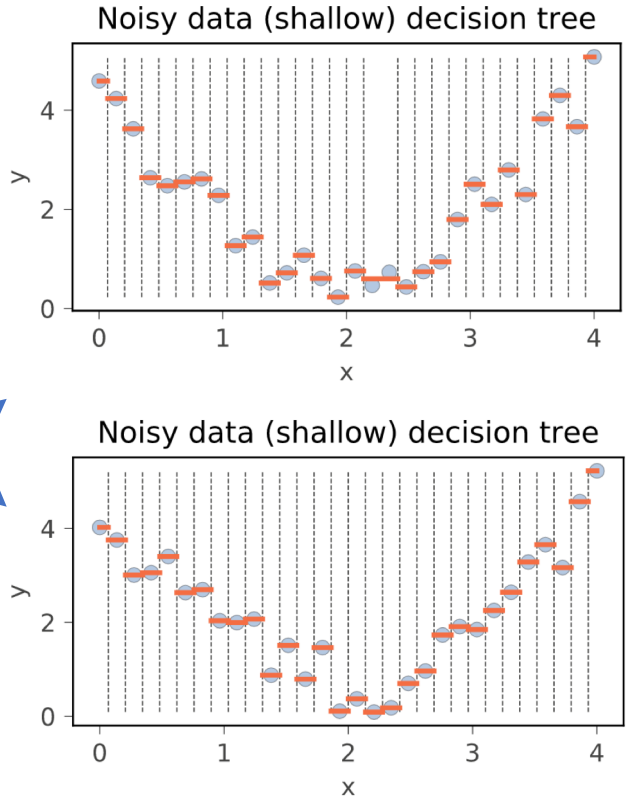
RF motivation

Leo Breiman (1996) introduced bagging
then Random Forests (2001)
https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf

- Decision trees can often get training errors close to zero because we can grow very large trees to partition the feature space into tiny regions with 1 or just a few observations/samples; trees have very **low bias**
- The downside is that decision trees overfit or, using stats nerds terminology, decision trees have **high variance** and don't generalize well
- Goal: keep the low bias, but increase the generality (reduce the variance)

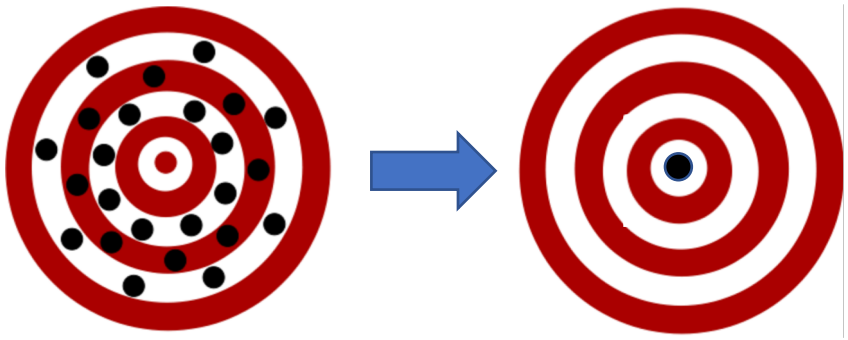
Variance is the key word here

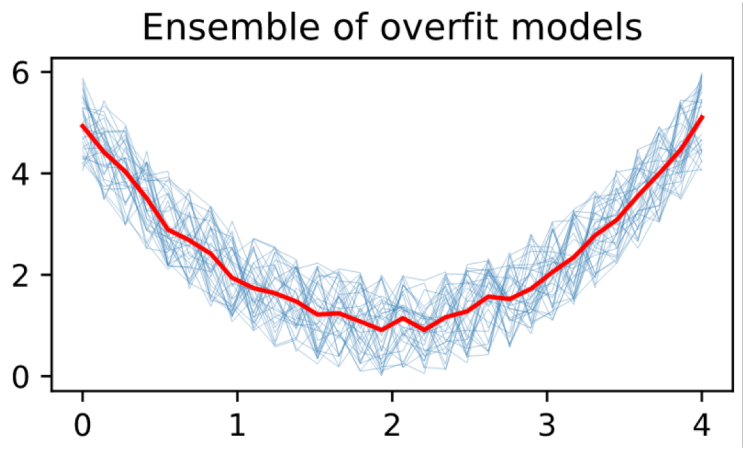
- Decision trees are sensitive to quirks in the data and so similar but different training sets drawn from the same distribution can yield very different models (parameters)
- (That's the definition of model variance)
- Ex: two i.i.d. X training sets fit by regr tree
- The predictions made by decision trees trained on i.i.d. sets bounce around, but their average approaches the correct value (same for i.d. sets)



Ensemble of high-variance decision trees

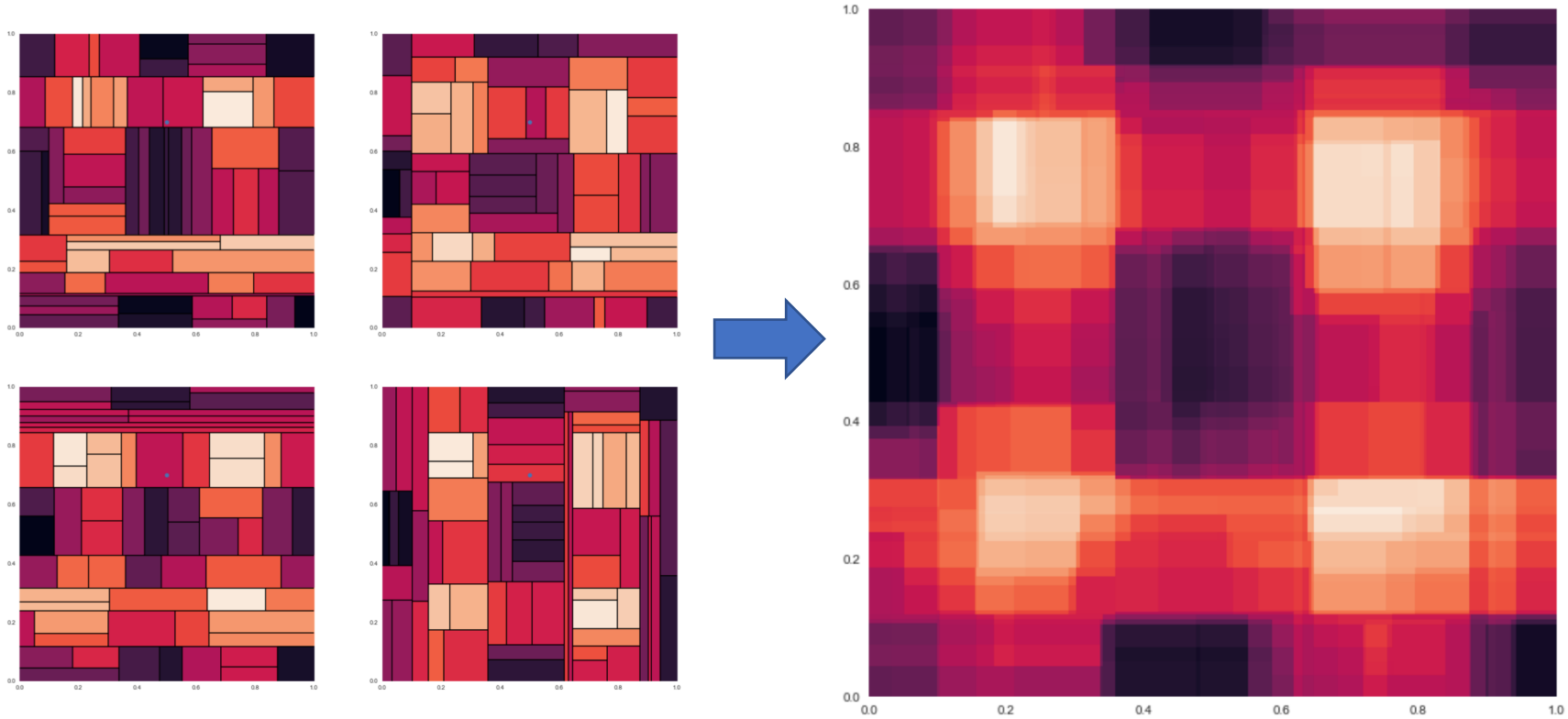
- Decision trees are inconsistent but are accurate on average across multiple identically distributed (i.d.) training sets
- Pretend we do have multiple i.d. X's and multiple fit models
- Then, we can average predictions from multiple models to get an accurate prediction:





Target image credit: http://ficscience.blogspot.com/2011/02/technicalities-of-technical-terms.html

RF: Overlapping feature space partitions



Images generated by USF MSDS alumnus Tyler White
https://gist.github.com/tylerwx51/fc8b316337833c877785222d463a45b0

Ensemble's effect on bias and variance

- Train T trees on T identically distributed X data sets (doesn't have to be independent)
- The average of the tree predictions is the same as the expected prediction from any tree trained on one of the X
- Central limit theorem says that if variance of *T* i.i.d. random variables is σ^2, the variance of the average is $\frac{\sigma^2}{T}$
- So, as we add trees, the variance of the prediction (average from ensemble trees) tightens towards true prediction!
- Problem: we only have one X training data set

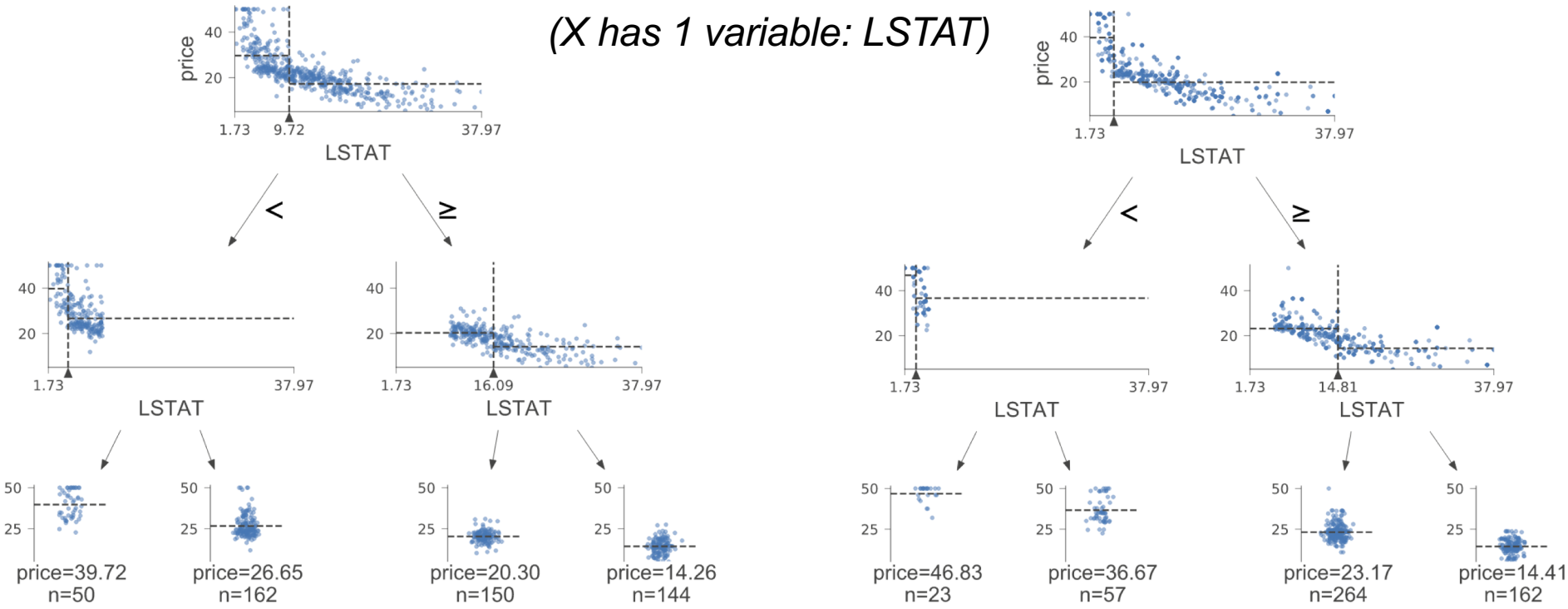
Bagging (Bootstrap aggregation)

- We only have one training set, X, but can *bootstrap* to get as many as we want from the same distribution
- These bootstrapped sets are i.d. but not independent (some overlap)
- *Bootstrap*: from *n* records, randomly select *n* with replacement
- Each bootstrapped data set will have about 2/3 of the unique overall records
- *Aggregation*: average the ensemble predictions (majority vote for classification)
- Analogy: real estate agents sent out to sample NYC apartments/prices; there will be some overlap but average of all agents' predictions is more accurate than an individual's

Bootstrapping gives slightly different trees

```
df = df.sample(len(df), replace=True) # boston data set
t = DecisionTreeRegressor(max_depth=2)
t.fit(X,y)
```

(X has 1 variable: LSTAT)



Problem: bootstrapping is i.d. not i.i.d.

- If just identically distributed and not necessarily independent, with positive pairwise correlation ρ between trees, then variance of the average for random vars is: (Eqn 15.1 ESLII Hastie *et al*):

$$\rho\sigma^2 + \frac{1-\rho}{T}\sigma^2$$

- As *T*, (num trees) grows, 2nd term disappears but correlation of trees would still have variance proportional to ρ (proportional to σ^2, variance of any single tree predictor)
- ELSII: "…the size of the correlation of pairs of bag trees limits the benefits of averaging."
- So, bagging helps to reduce variance but still has correlation ρ>0

What does “correlated trees” mean?

- Model parameters are correlated; “not independent” better term
- **Analogy**. Real estate agents sampling bias: if they visit apartments they observe other agents visit, variance might not shrink with more agents; worst-case: they all visit the same apartments; best case is they choose all apts independently
- **Worst-case**. Imagine replicating single decision tree *T* times, variance of average prediction would be same as variance of single tree (i.e., bad strategy); if ρ=1, variance of avg = σ^2
- **Best-case.** if ρ=0, trees are uncorrelated and variance of average converges to 0 as T grows

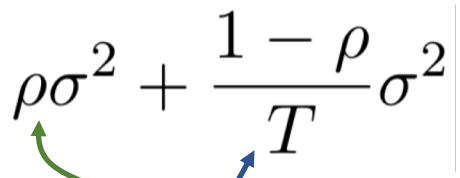
$$\rho\sigma^2 + \frac{1-\rho}{T}\sigma^2$$

Bagging summary

- Adding bagged trees does not increase **bias**; the avg of tree predictions is same as the expected prediction from any single bootstrapped tree
- Keep in mind, however, that a tree fit to bootstrapped data is only using 2/3 of the data and so each bagged tree will have higher bias than single decision tree fit to entire data set
- Bagging does reduce **variance**, it is averaging predictions from lots of non-identical models afterall
- Degree of tree similarity, ρ, influences ability to reduce variance to 0
- If ρ=1, no reduction in variance, but ρ=0 means adding trees reduces variance to 0 asymptotically

$$\rho\sigma^2 + \frac{1-\rho}{T}\sigma^2$$

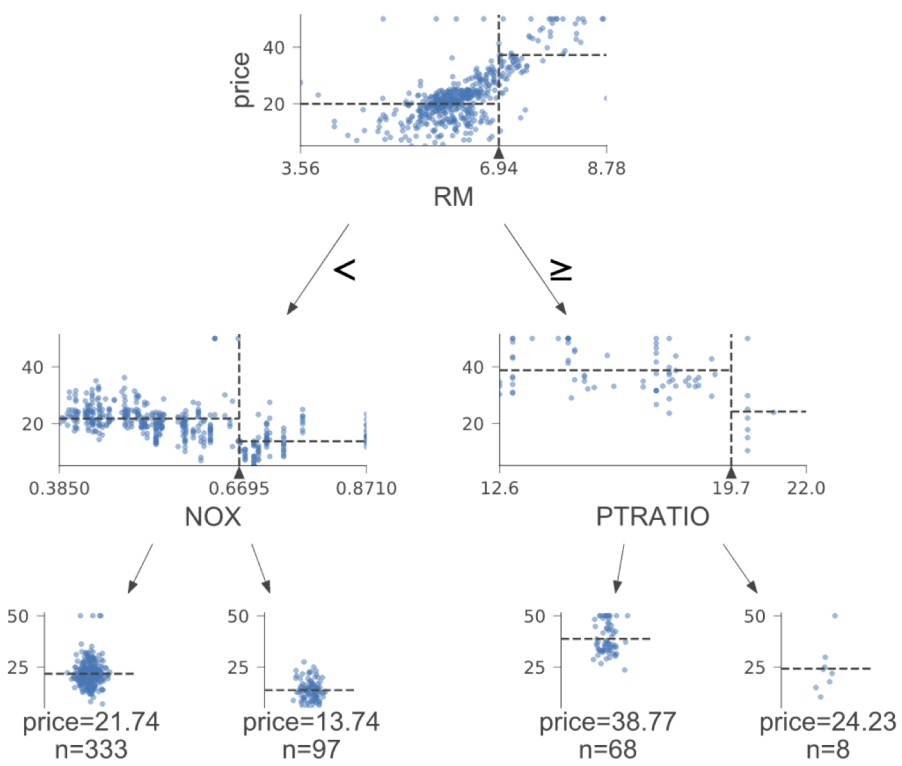
De-correlating trees to get Random Forests (RFs)

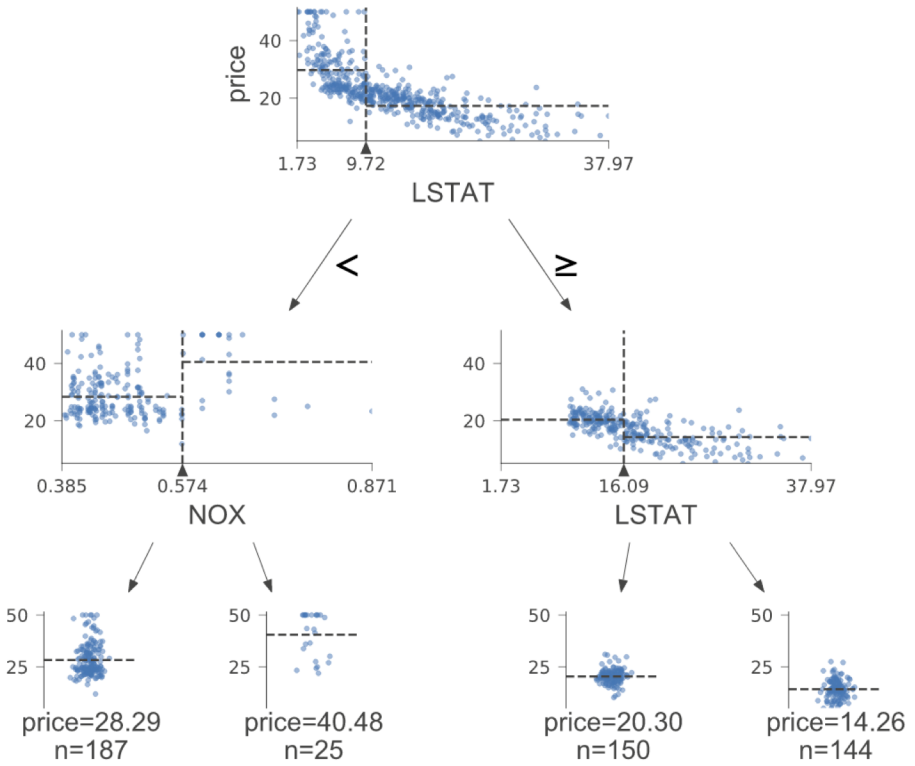


- Bagging overcomes most of the overfitting (Hastie *et al* p589 say ρ usually small like 0.05)
- Can do a little better de-correlation of the trees to reduce ρ
- Restrict the available features at each split (random selection of m)
- Choose max features per split, m <= p, such as sqrt(p)
- Imagine one strongly predictive var out of p, then all trees would be similar; initial root splits, and many others, will likely be same
- Make sure chance of selecting predictive variables (m/p) is high enough to find them (See ESLII p596)
- Let validation error be your guide to m (and other hyperparameters)

(Boston data)

Max features = 5 (of 13) gets diff trees

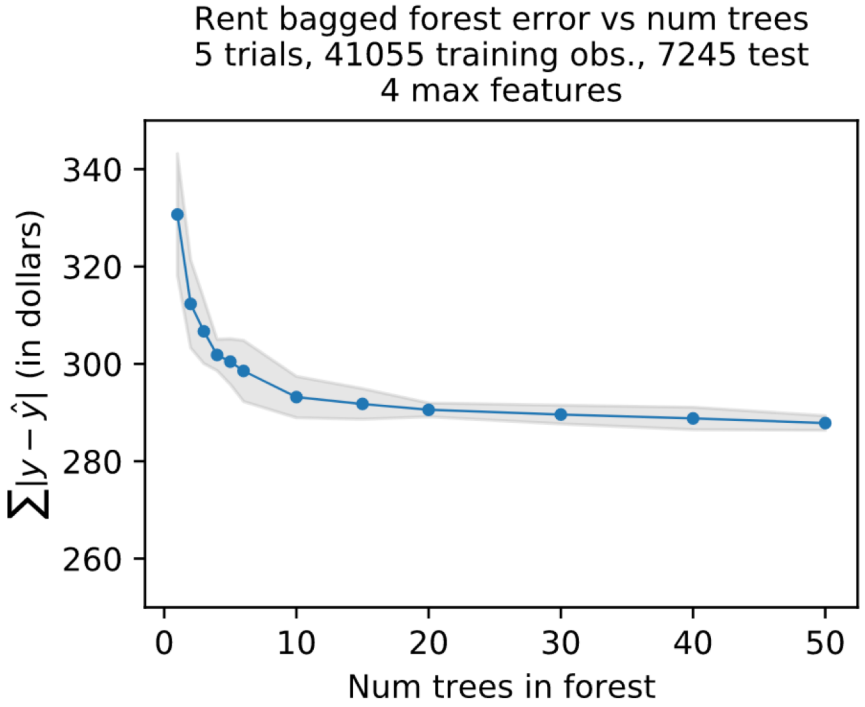


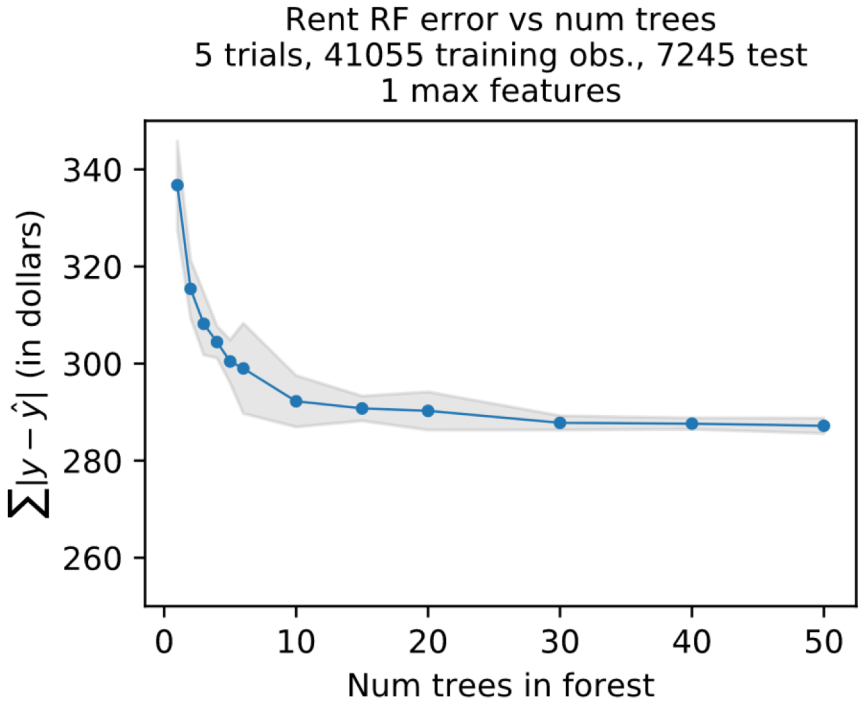


Choose 5 randomly selected features during EACH split

Compare bagged trees vs RF (Rent)

Very little difference with just p=4 features

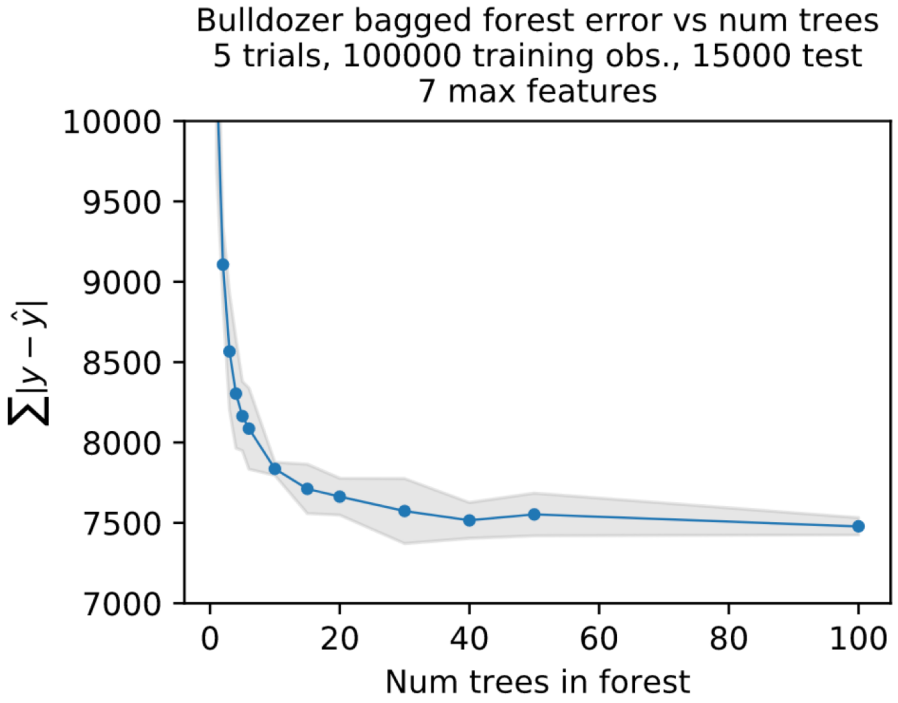


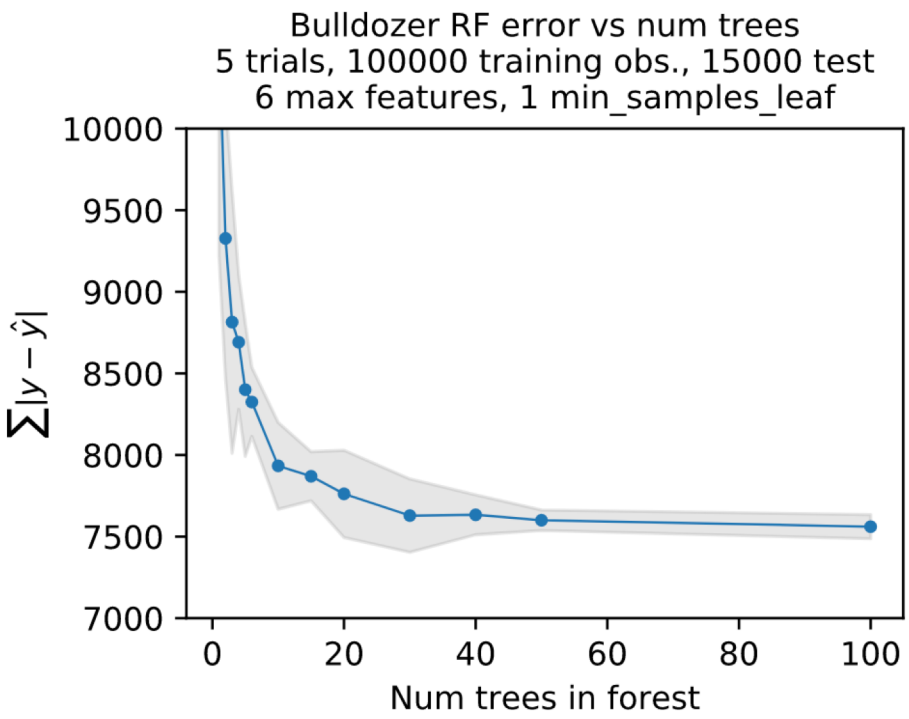


Some data sets will be more sensitive to max features than others

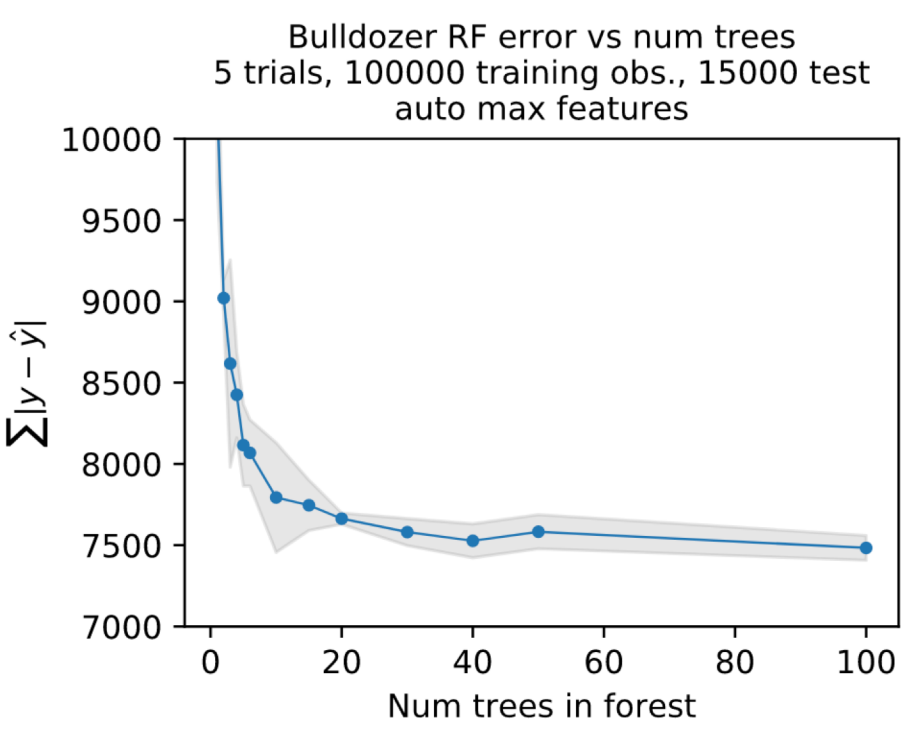
Compare bagged trees vs RF (bulldozer)

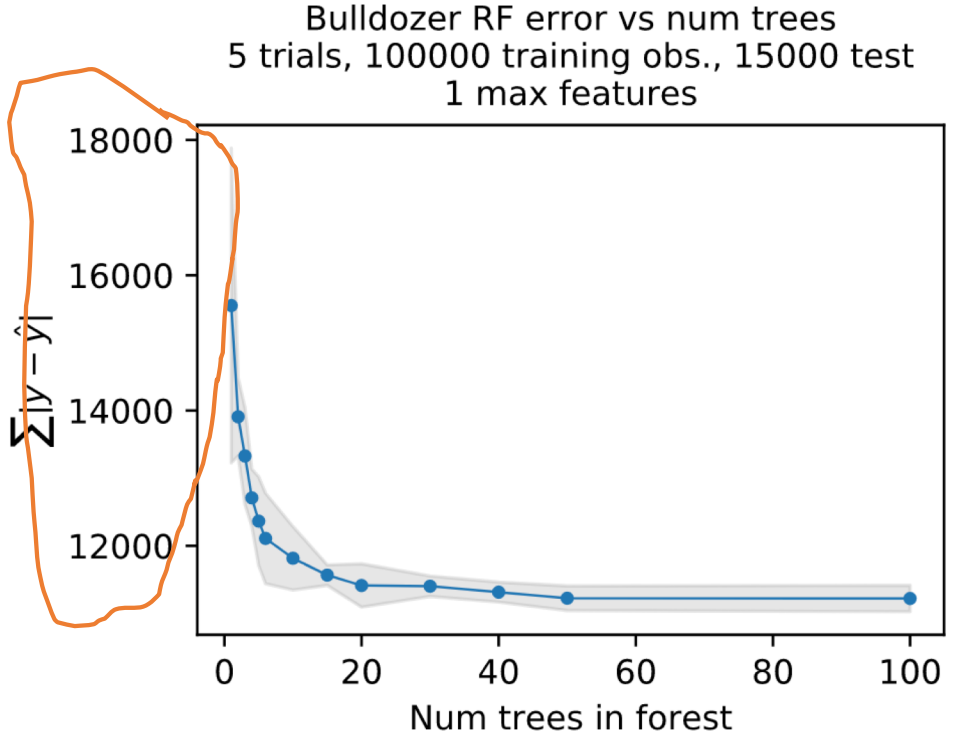
For this data, using all p=7 is best





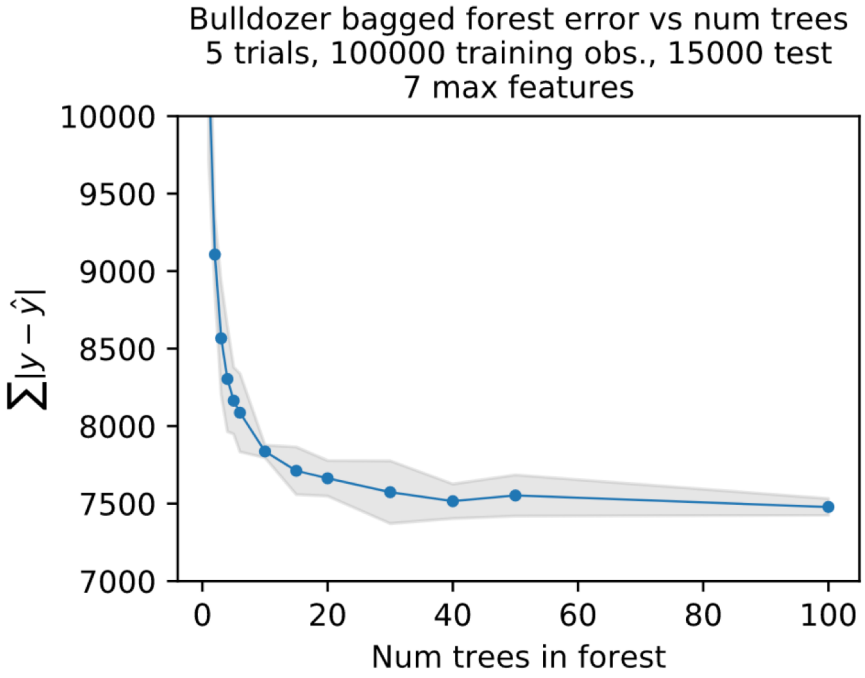
If max_features too low, high bias



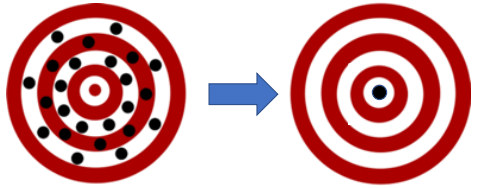


Effect of forest size on bias

- Why does accuracy improve greatly (initially) as we add trees?
  - Each tree sees only 2/3 of data so adding bootstrapped trees increases use of training data
- Why does accuracy asymptotically approach a minimum instead of continual improvement?
  - With enough trees, ensemble sees 100% of the training data; it's approaching the bias of single decision tree in ideal world where it's not overfit



Properties (Breiman 2001)



- p4 "*Random forests do not overfit as more trees are added*" **Why**?
  - Adding more trees actually REDUCES variance, and overfitting==variance
  - New trees get averaged in so each new tree has less individual effect
  - New trees balance each other out, one might be too high, another too low
- p7 "*It's relatively robust to y outliers and X noise*" **Why**?
  - y outliers get shunted to their own leaf since doing so reducing loss function, particularly if squared-error is used
  - Noise vars aren't predictive so not chosen as split vars
- p10 Bagging helps more the more unstable the model. **Why**?
  - Averaging is a smoothing operator, squeezing predictions to centroid
  - If model is low variance already, there is no point in bagging

Properties continued

- RFs are scale and range insensitive in features and target y **Why**?
  - Comparing feature values in decision nodes not doing math on them
  - Computing mean or mode of y to predict
- ESLII p596 "*Classifiers are less sensitive to variance [than regressors]*" **Why**?
  - (not sure haha) I believe it has something to do with mode vs mean (mode is same until a threshold whereas mean is influenced by any value added, unless it is also the mean)

Bootstrapping vs subsampling

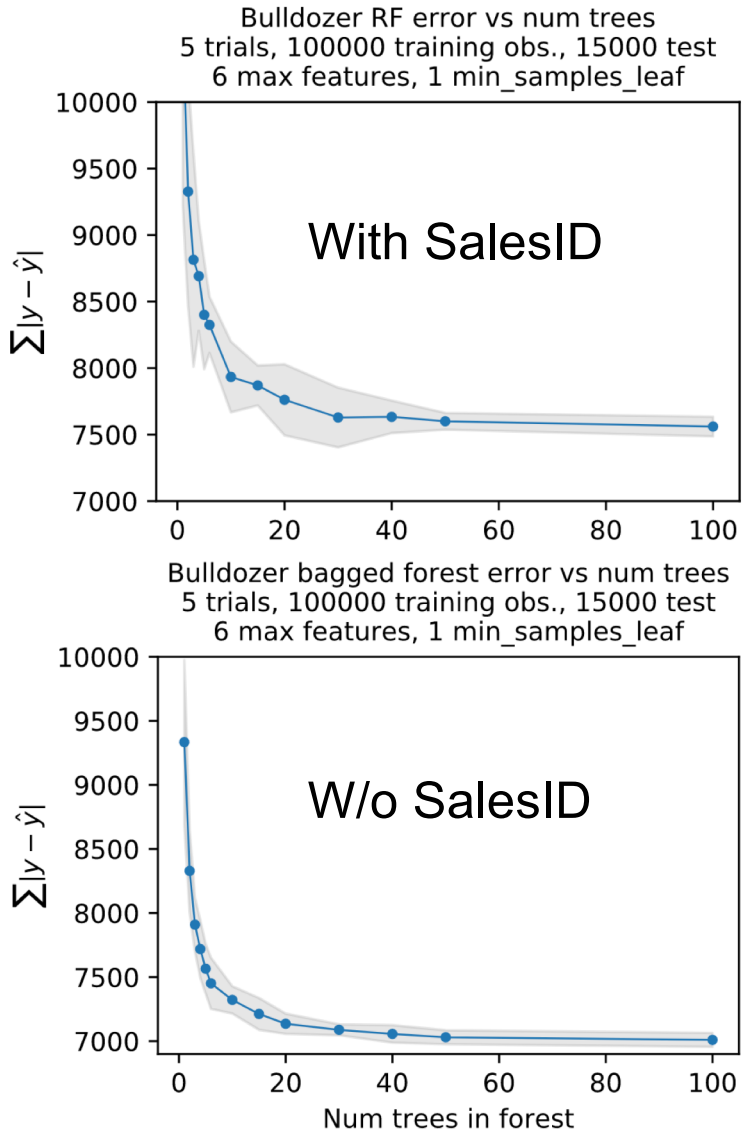
- Bootstrapping is sampling with replacement vs subsampling w/o replacement
- 😲 Friedman and Hall (2000): subsampling also works, showing that training trees with n/2 subsamples is similar in bias/variance to bagging http://statweb.stanford.edu/~jhf/ftp/bag.pdf
- Smaller training set is a big win in terms of speed
- Using even smaller fractions of n improve generality (reduce variance) because trees are less correlated (they work on different data chunks); note that each tree would become more biased as n subsample size decreases

RF Tuning strategy

- Good news: very little tuning needed
- Start with maybe 20 trees and work upwards til validation error stops getting better; or just pick 100
- Sklearn uses max_features= sqrt(p) by default; try dropping this using log(p), or similar; ESLII suggests p/3 for regression and sqrt(p) for classification
- Try adjusting min samples per leaf: 1, 3, 5, 10, 25, 100
- Goal: minimize validation error
- Can also try grid search, but I never bother;
  Start with num trees, then tune the others

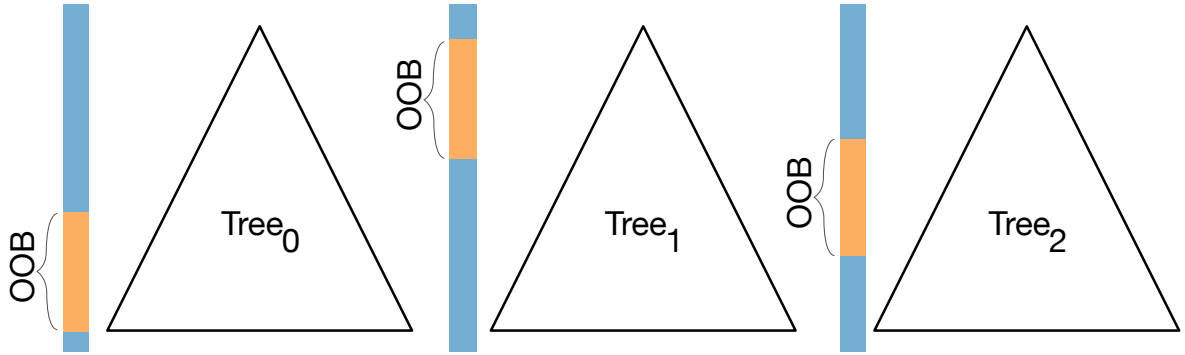
Feature engineering beats model tuning

- SalesID: unique record ID, and is never seen again in future predictions
- Is that useful for prediction? No
- Does the model think it's useful? Yes
- Model is overfit not on noise but on falsely-predictive feature
  - Could be that sales ID correlates with inflation or change in type of models sold in auction creates "trend" in sale prices
- A case where using LESS data improves the model a lot ($500 diff)
- Dropping useless features also often gives a small bump

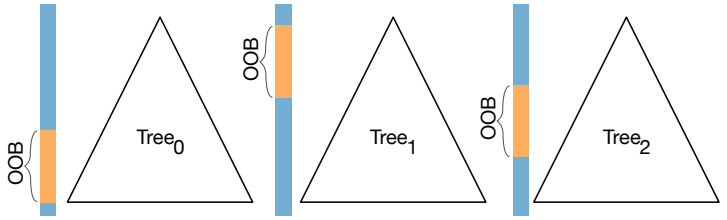


RF Out-of-bag (OOB)

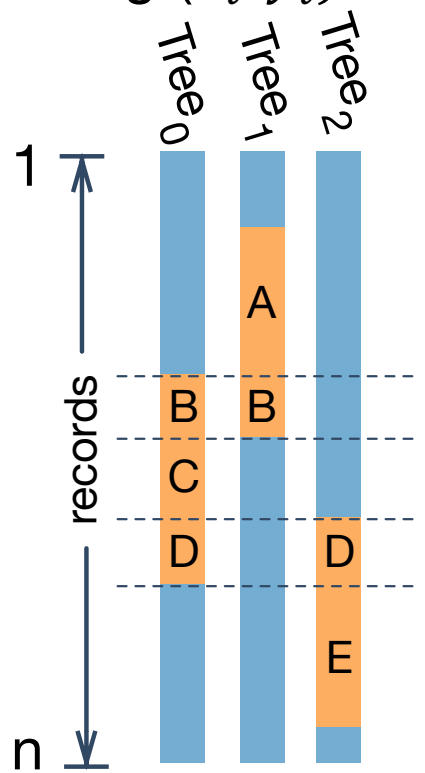
- RFs have a major advantage over other models: OOB metrics
- Each tree is trained on 63% of data, leaving 37% OOB
- OOB subsamples available to the trees are different
- It's an excellent estimate of the validation error
- Stick with OOB unless default sklearn score() is not suitable
- Not having to process training and validation sets separately is a huge productivity win (assuming significant feature engineering)



Computing OOB predictions



- Get $\hat{y}_i$ by averaging estimates from trees not trained using (x_i, y_i)
  - Image to right; blue is training set, OOB orange
  - Trees from same labeled region of $x^{(i)}$ give $\hat{y}_i$
  - Must find all trees not trained on $x^{(i)}$
  - E.g., compute $\hat{y}_i$ for B region using Trees 0, 1
  - No OOB error estimate is possible for unlabeled regions
- Do not make predictions for OOB and compute error per tree!
- Each tree has high bias, so OOB scores from 1 tree'd be very high
- Must compute metric on predicted $\hat{y}$ vector as usual
- Algorithms for regression and classification later



OOB continued

- OOB error might slightly overestimate test set error. Why?
  - OOB samples are not predicted with all trees in forest whereas test set uses whole forest, which presumably has lower bias/variation [1]
- Some research suggests OOB overestimates for binary classification https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0201904
- OOB metrics don't affect training, just gives metric
- Compare OOB with validation set:
  - If we add (predictive) feature and OOB is still ok, but the validation set is worse, the validation set is not good; e.g., different distribution or time sensitive
- OOB not to be used with time-sensitive data sets. Why not?

[1] For n<<p case, see paper https://file.scirp.org/Html/9-1240025_8072.htm

Extremely randomized trees (Geurts *et al* 2006)

- The variable/value pair is highly sensitive to the training set, and responsible much of error rate
- "*The optimal cut-point was shown to depend very strongly on the particular learning sample used…this cut-point variance appeared to be responsible for a significant part of the error rates of tree-based methods.*" https://link.springer.com/article/10.1007/s10994-006-6226-1
- Geurts wondered if more randomness could reduce variance further
- Pick random split value in min(X[:,j]) .. max(X[:,j]), ignoring y!
- Like RF, select m<=p variables and choose var/value with lowest loss
- Fits using entire X training set, not bootstrap and not subsample (trying to reduce bias)
- Our use of just 11 (not n) X candidate values in the project is similar (an effort to reduce variance and increase speed)

The RF algorithms

Fitting RFs

Algorithm: $RFfit(X,\ y,\ loss,\ ntrees,\ max\_features,\ min\_samples\_leaf)$

for $i = 1..ntrees$ **do**

$X', y' = bootstrap(X, y, size = |X|)$

$T_i = RFdtreefit(X', y', loss, max\_features, min\_samples\_leaf)$

end

For regression, pass in loss = MSE or stddev
For classifier, pass in loss = gini

Fitting a single tree in RF

Same as decision tree except we pass max_features to RFbestsplit()

Algorithm: $RFdtreefit(X,\ y,\ loss,\ max\_features,\ min\_samples\_leaf)$

if $|X| \leq min\_samples\_leaf$ **then** return Leaf(y)
$col, split = RFbestsplit(X, y, loss, max\_features)$
if $col = -1$ **then** return Leaf(y)
$lchild = RFdtreefit(X[X_{col} \leq split], y[X_{col} \leq split], ...)$
$rchild = RFdtreefit(X[X_{col} > split], y[X_{col} > split], ...)$
return $DecisionNode(col, split, lchild, rchild)$

Finding best split in decision node in RF

Algorithm: *RFbestsplit(X, y, loss, max_features)*

$best = (col = -1, split = -1, loss = loss(y))$
$vars =$ pick $max\_features$ variables from all p
for $col \in vars$ **do**
  $candidates =$ randomly pick $k \ll n$ values from X_{col}
  foreach $split \in candidates$ **do**
    $yl = y[X_{col} \leq split]$
    $yr = y[X_{col} > split]$
    if $|yl| < min\_samples\_leaf$ *or* $|yr| < min\_samples\_leaf$ **then continue**
    $l = \frac{|yl| \times loss(yl) + |yr| \times loss(yr)}{|y|}$ $(weighted\ average\ of\ subregion\ losses)$
    if $l = 0$ **then return** $col, split$
    if $l < best[loss]$ **then** $best = (col, split, l)$
  end
end
return *best[col], best[split]*

Only diff with decision tree

Pick, say, 11 not all possible X values. We get better generality and code is much faster!

Should pick midpoint between split value and next smallest X

Simplest RF prediction (ESLII p588)

- But doesn't use all information to make best prediction
- Should use weighted averages / votes

Regression: $\hat{f}_{\mathrm{rf}}^{B}(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\mathrm{rf}}^{B}(x) = \textit{majority vote } \{\hat{C}_b(x)\}_1^B$.

RF prediction

Weighted average of y values among the leaves reached by running x down each tree

Algorithm: $RFpredict_{regr}(\{T_1..T_{ntrees}\},\ x)$

Let $leaves = \{leaf(T_t, x)\ \forall t = 1..ntrees\}$

$nobs = \sum_{t=1}^{ntrees} |leaves_t|$

$ysum = \sum_{t=1}^{ntrees} \sum_{y \in leaves_t} y$

return $\frac{1}{nobs} ysum$

Count all y votes among the leaves reached by running x down each tree

Algorithm: $RFpredict_{class}(\{T_1..T_{ntrees}\},\ x)$

$counts[k] = 0\ \forall$ classes k

foreach $t = 1..ntrees$ **do**

  $leaf = leaf(T_t, x)$ $\quad (leaf\ reached\ by\ x)$

  foreach $y \in leaf$ **do**

    $counts[y]$ += 1 $\quad (track\ count\ of\ leaf\ modes)$

  end

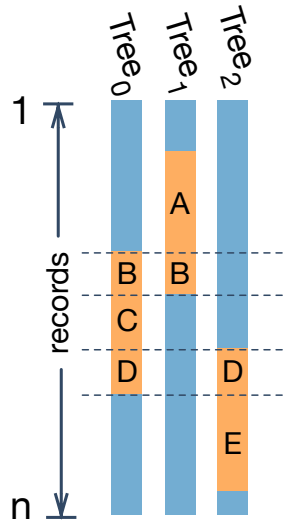
end

return $\text{argmax}(counts)$

OOB regression scoring

Algorithm: $oob\_score_{regr}(RF,\ X,\ y)$

Let $oob\_counts[i] = 0\ \forall$ records $i = 1..|X|$ $(Num\ obs.\ in\ all\ leaves\ reached\ by\ X[i])$
Let $oob\_preds[i] = 0\ \forall$ records $i = 1..|X|$ $(Predictions\ for\ X[i]\ weighted\ by\ leaf\ size)$
foreach $t \in RF$ **do**
  $leafsizes = |t.leaf(X[t.oob])|$ $(Num\ samples\ in\ leaf\ reached\ by\ each\ X)$
  $oob\_preds[t.oob]\ +\!= leafsizes \otimes t.predict(X[t.oob])$
  $oob\_counts[t.oob]\ +\!= leafsizes$
end
$oob\_avg\_preds = \frac{oob\_preds[oob\_counts>0]}{oob\_counts[oob\_counts>0]}$
return R^2 $score\ for\ (y[oob\_counts > 0], oob\_avg\_preds)$

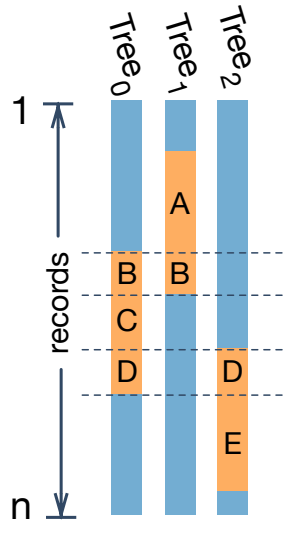


Assumes each tree collects OOB sample indexes during fit()

OOB classification scoring

Algorithm: $oob\_score_{class}(RF,\ X,\ y)$

Let $oob\_counts[i] = 0\ \forall$ records $i = 1..|X|$ $(Num\ trees\ w/predictions\ for\ X[i])$
$(Create\ 2D\ matrix\ tracking\ vote\ counts\ per\ class\ for\ each\ X[i])$:
Let $oob\_preds[i,k] = 0\ \forall$ records $i = 1..|X|, k = 1..|unique(y)|$
foreach $t \in RF$ **do**
  $leafsizes = |t.leaf(X[t.oob])|$ $(Num\ samples\ in\ leaf\ reached\ by\ each\ OOB\ X)$
  $tpred = t.predict(X[t.oob])$
  $oob\_preds[t.oob, tpred]\ +\!= leafsizes$ $(count\ weighted\ class\ votes)$
  $oob\_counts[t.oob]\ +\!= 1$ $(track\ num\ trees\ used\ for\ each\ OOB\ X)$
end
for $i\ such\ that\ oob\_counts[i] > 0$ **do**
  $oob\_votes[i] = \arg\max_k oob\_preds[i,k]$
end
return $accuracy\ of\ y[oob\_counts > 0] = oob\_votes$



Assumes each tree collects OOB sample indexes during fit()