Random Forest
Supervised model
really an 'ensemble of decision trees'

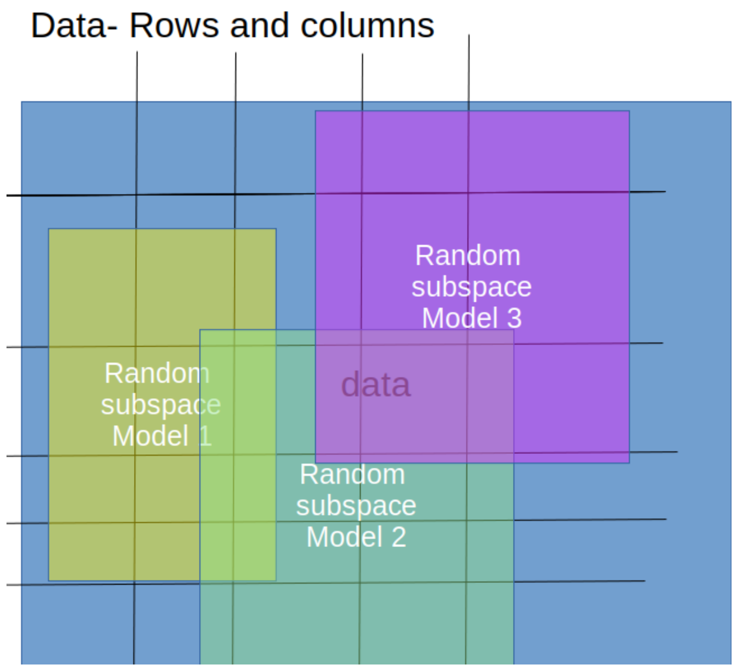
- fast
- flexible
- dont have many parameters to tune
- can use to predict a number (like linear egression)
- can use to predict a class (like logistic regression)

Rest on idea of "if you have a model that's terrible, but is better than random, and if you can make thousands of these models that are all terrible <u>in different ways (means they are not corelated with each other)</u>,then the average of those 1000's of models will be very powerful'
kind of like the wisdom of crowds

INTUITION: if you have 1 model thats terrible in that it always predicts low, then you likely have another thats terrible in that it predicts high, they average out.

Its a random forest because each of the models we create samples a <u>random subspace</u> of the data.

Random subspace
using some of the columns and some of the rows
the rows and columns are NOT contiguous
these are all randomly sampled (both rows and columns)



Building the trees
Each tree trained on random subspace

When building the trees, each time a split in a tree is considered, a random sample of m predictors is chosen as split candidates from the full set of p predictors.

The random rows and columns and selection of random features to eval at each split point serves to make each tree less corelated with every other tree.
Predictions
For each of these random subspaces build a decision tree

age	sex	issmoker	lungcapacity
30	M	Y	44
:			

```
dec tree 1
   issmoker
 Y             N
30             70
               age<18
       Y              N
      20              50
```

make a bunch of these models using different rows and columns

Predict: 1 may predict 50, 2nd 30 3rd 20 ….j
Predictions from the trees are averaged across all decision trees resulting in better performance than any single tree in the model.

Regression: Prediction is the average prediction across the decision trees.

Classification: Prediction is the majority vote class label predicted across the decision trees.

To increase randomness (to avoid correlations) make your random subspaces smaller and have more trees. Each tree is worse (less powerful) but when you average, you get a better prediction. But takes longer than building a few trees on larger subspaces.

Want to be even more random?   Use extremely randomized trees (see ExtraTreesClassifier and ExtraTreesRegressor classes).   Instead of splitting on the perfect point for continuous vars (or other vars), select a few points withen range and randomly choose one. You dont get the best information gain, but you avoid the problem where you are splitting on the same number for every tree which leads to higher correlations between trees.

If highly imbalanced then maybe you want to do stratified sampling to ensure you have minority class representated.

How to find the optimal size subspace, use fewer rows and columns and more trees so you dont overfit

As you add trees it gets better at predicting
probably want sampling without replacement for the rows you pull

Out of Sample data
Say we train 100 trees, each tree uses a different random subspace. You can use the rows that are NOT in this subspace as a validation set

Row	Tree1	Tree2	Tree3	Tree4	.....	Treen	OOB
1	X	O	O	X……			19.8
2							20
3							:
4							
:							
N							

X=not used to train   O=Used to train

So when testing row 1, I use every tree that did NOT
Train on row 1 (Tree2, Tree3…) to make a prediction
That's called Out Of Band estimate for row 1
Do this for every row, use to get AUC or Log Liklyhood or whatever

Dont prune or weight or select or you are introduce bias

Hyperparameters

Depth
Deeper trees are often more overfit to the training data, but also less correlated, which in turn may improve the performance of the ensemble. Depths from 1 to 10 levels may be effective.

Number of Trees

Mtry what to set to

A good heuristic for regression is to set this hyperparameter to 1/3 the number of input features.

•num_features_for_split = total_input_features / 3

> *For classification problems, Breiman (2001) recommends setting mtry to the square root of the number of predictors.*

— Page 387, Applied Predictive Modeling, 2013.

A good heuristic for classification is to set this hyperparameter to the square root of the number of input features.

•num_features_for_split = sqrt(total_input_features)