decision forests

Decision Forests are another type of ensemble method where weak classifiers are combined to build a stronger algorithm.

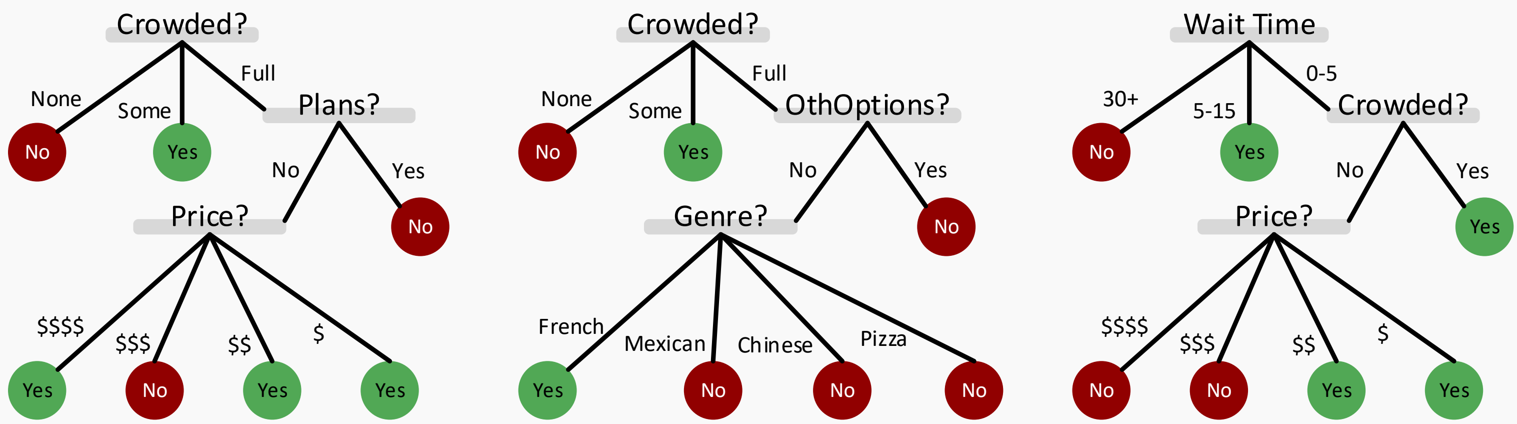
- **Decision Forests** are a powerful prediction tool that are highly complex
- They are another type of **black-box** algorithm where interpretability is low
- Similar logic to **Boosted Decision Trees**; averaging many uncorrelated, yet accurate, models reduces overall variance

Returning to the illustrative restaurant scenario from the Decision Tree example:

Example: Will the customer wait for a table at a restaurant?

- OthOptions: Other options, True if there are restaurants nearby.
- Weekend: This is true if it is Friday, Saturday or Sunday.
- Area: Does it have a bar or other nice waiting area to wait in?
- Plans: Does the customer have plans just after dinner?
- Price: This is either $, $$, $$$, or $$$$
- Precip: Is it raining or snowing?
- Genre: French, Mexican, Thai, or Pizza
- Wait: Wait time estimate: 0-5 min, 6-15 min, 16-30 min, or 30+
- Crowded: Whether there are other customers (no, some, or full)

	OthOptions	Weekend	Area	Plans	Price	Precip	Genre	Wait	Crowded	Stay?
x_1	Yes	No	No	Yes	$$$	No	French	0-5	some	Yes
x_2	Yes	No	No	Yes	$	No	Thai	16-30	full	No
x_3	No	No	Yes	No	$	No	Pizza	0-5	some	Yes
x_4	Yes	Yes	No	Yes	$	No	Thai	6-15	full	Yes
x_5	Yes	Yes	No	No	$$$	No	French	30+	full	No
x_6	No	No	Yes	Yes	$$	Yes	Mexican	0-5	some	Yes
x_7	No	No	Yes	No	$	Yes	Pizza	0-5	none	No
x_8	No	No	No	Yes	$$	Yes	Thai	0-5	some	Yes
x_9	No	Yes	Yes	No	$	Yes	Pizza	30+	full	No
x_{10}	Yes	Yes	Yes	Yes	$$$	No	Mexican	6-15	full	No
x_{11}	No	No	No	No	$	No	Thai	0-5	none	No
x_{12}	Yes	Yes	Yes	Yes	$	No	Pizza	16-30	full	Yes



A new observation is applied to the models above:

Genre: Mexican
Price: $$
Crowded: Full
Wait Time: 5-15
Plans: No
OthOptions: No

Applying the observation to the models above, the labels respectively assigned are **yes**, **yes**, **no**.

Therefore, the combination of the models apply the majority vote as the predicted label: **yes**

More precisely, the method utilizing **bootstrapping** of samples. A **bootstrap sample** of size n: A sample of n points drawn with replacement at random from the training data. Each point is taken one at a time and then replaced prior to selecting another. Thus some points will be repeated as expected.

Decision Forest Pseudocode

For $t = 1{:}T$

Draw a **Bootstrap Sample** of size n from the training dataset

Build a **Decision Tree** $\mathbf{tree}_t$ using the splitting and stopping procedure:

Choose m features at random (out of p)

Evaluate the splitting criteria on all features m

Split on the most valuable feature

Repeat until the node has $< n_{\min}$ observations, then cease splitting

Output all trees (noting that only no pruning takes place, only splitting)

To predict on a new observation x, the majority vote of the trees is applied to x

Decision Forests Compared to Decision Trees

- **Decision Trees** use all sample data; **Decision Forests** use bootstrapped resamples to compute many decision trees.
- **Decision Trees** are allowed to use all of the features for consideration of each split; **Decision Forests** only consider m randomly chosen features for each split.
- **Decision Forests** do not utilize pruning, making trees fit more tightly and reduce bias.
- **Decision Forests** apply majority vote to make predictions; **Decision Forests** use a single tree.

Measuring Variable Importance with Decision Forests

In measuring the importance of feature $\boldsymbol{j}$:

The data **not used** to construct $\mathbf{tree}_t$ is taken: referred to here as "out-of-bag" $\mathbf{OOB}_t$

The $\mathbf{error}_t$ will be computed on the $\mathbf{OOB}_t$ data using the model $\mathbf{tree}_t$

The $\mathbf{OOB}_t$ data is then taken and randomly permutated on the $\boldsymbol{j^{th}}$ feature values:

$$\mathbf{OOB_t} \rightarrow \begin{bmatrix} x_{11} & \boldsymbol{x_{12}} & x_{13} \\ x_{21} & \boldsymbol{x_{22}} & x_{23} \\ x_{31} & \boldsymbol{x_{32}} & x_{33} \end{bmatrix} \rightarrow \text{randomly permuted} \rightarrow \begin{bmatrix} x_{11} & \boldsymbol{x_{32}} & x_{13} \\ x_{21} & \boldsymbol{x_{12}} & x_{23} \\ x_{31} & \boldsymbol{x_{22}} & x_{33} \end{bmatrix} \rightarrow \mathbf{OOB}_{t,\mathbf{permuted}}$$

The logic behind a random permutation is explained hypothetically: if the feature $\boldsymbol{j}$ was unimportant originally, randomly arranging the feature values would have little impact.

Then, the $\mathbf{error}_{t,\mathbf{permuted}}$ will be computing using model $\mathbf{tree}_t$ on data $\mathbf{OOB}_{t,\mathbf{permuted}}$

The "**raw importance**" of variable $\boldsymbol{j}$ is thus the average over trees of the difference:

$$\frac{1}{T}\sum_{\text{trees } t}\left(\text{error}_t - \text{error}_{t,\text{permutated}}\right)$$

Decision Forests for Regression

For $\boldsymbol{t = 1{:}T}$

Draw a **Bootstrap Sample** of size $\boldsymbol{n}$ from the training dataset

Build a **Decision Tree** $\mathbf{tree}_t$ using the splitting and stopping procedure:

Choose $\boldsymbol{m}$ features at random (out of $\boldsymbol{p}$)

Evaluate the splitting criteria on all features $\boldsymbol{m}$

Split on the most valuable feature

Repeat until the node has $< \boldsymbol{n}_{\mathbf{min}}$ observations, then cease splitting

Output all trees (noting that only no pruning takes place, only splitting)

To predict on a new observation $\boldsymbol{x}$, the ~~majority~~ **average** vote of the trees is applied to $\boldsymbol{x}$

Summary of Decision Forests

Advantages

- Complex and powerful prediction tool that is highly nonlinear

Disadvantages

- Black-box algorithm that is difficult to interpret
- Prone to overfitting unless tuned carefully (not intuitive with R package)
- Slow running algorithm with high computational cost

Decision Forests are essentially a collection of overfitted heuristic models that are averaged over their individual majority votes to predict new labels in a dataset.