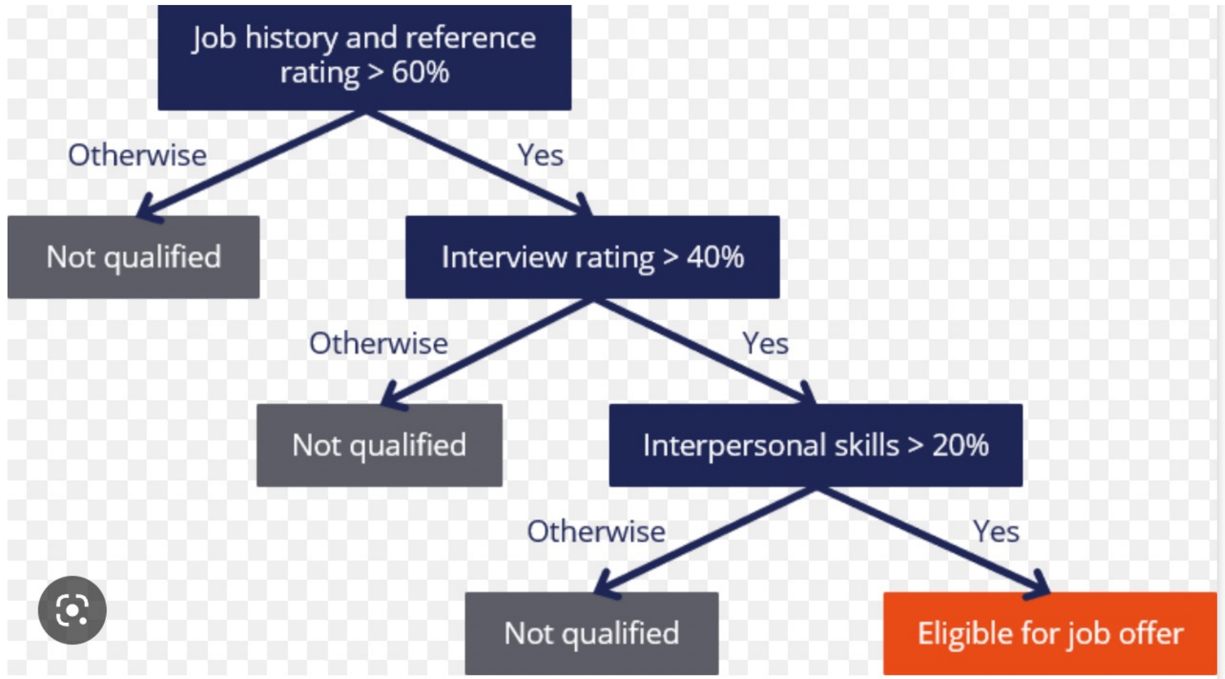
DATA 300

Topic 6: Tree-based methods

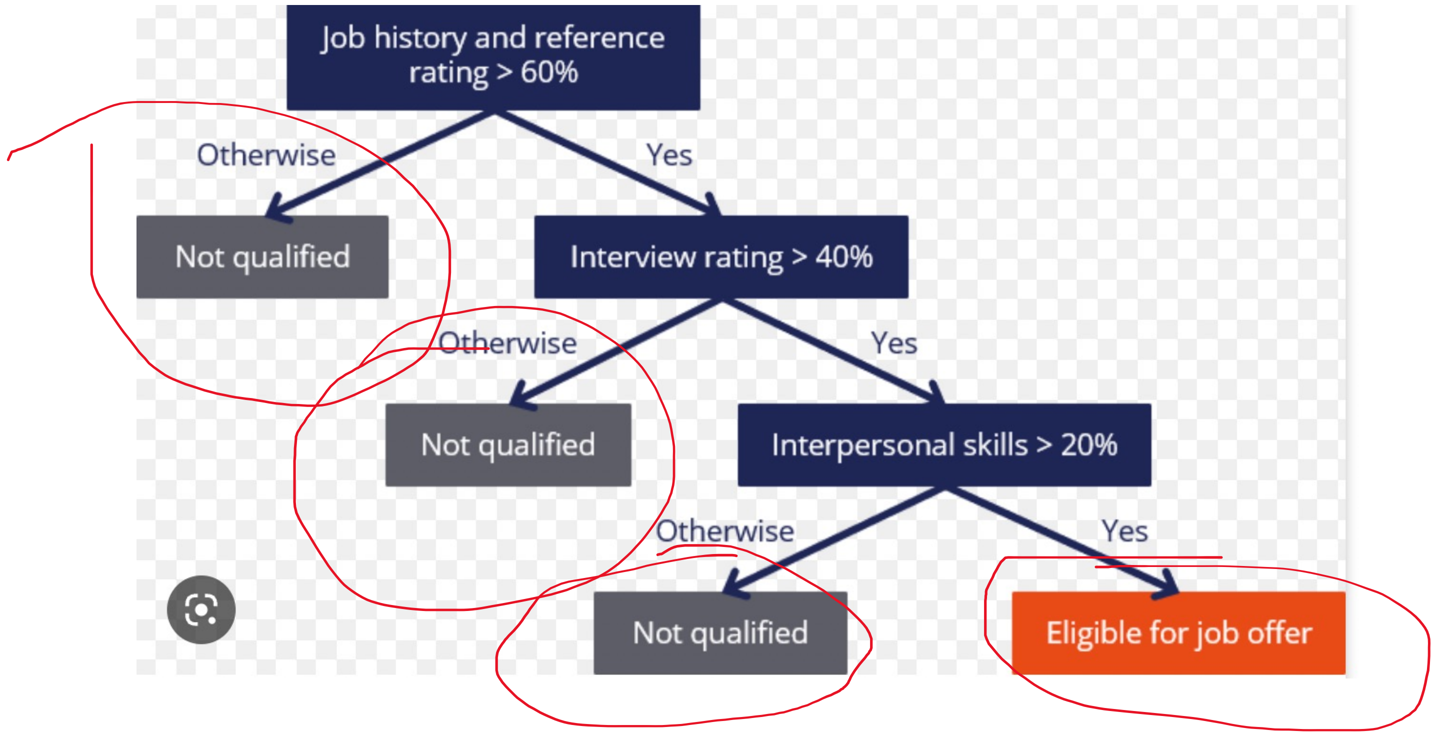
Agenda

- Decision trees
  - Overview
  - Regression trees
  - Tree pruning
  - Classification trees
- Tree-based methods

Overview of decision tree



“Regions” in decision trees



The circled ones are called *terminal nodes*, other nodes are called *internal nodes*, and the nodes are called being connected by *branches*.

Overview of decision tree

Theoretically, if we think of a decision tree as dividing the **predictor space** to multiple regions $R_1, \ldots, R_J$, each region has the exact same response value, then the regions should be divided such that

$$\text{minimize } \sum_{j=1}^{J} \sum_{i \in R_j} Error(y_i - \widehat{y_{R_j}}),$$

where

- Error() is some sort of an error function,
- $\widehat{y_{R_j}}$ is the predicted response value for all sample units in region R_j.

Regression tree

To construct a regression tree, we need to:

- at every step, identify a variable to create a split, or make it a final region.
- all splits should be constructed such that across all final regions, we could minimize Sum of Squared Residuals (no surprise):

$$\sum_{j=1}^{J} \sum_{i \in R_j} \left(y_i - \widehat{y_{R_j}} \right)^2$$

Regression tree

To construct a regression tree, we need to:

- at every step, identify a variable to create a split, or make it a final region.
- all splits should be constructed such that across all final regions, we could minimize Sum of Squared Residuals (no surprise):

$$\sum_{j=1}^{J} \sum_{i \in R_j} \left(y_i - \widehat{y_{R_j}} \right)^2$$

Thoughts on how to implement it? How to find these regions? Or can we even feasibly find them?

Search algorithm for decision trees

It is computationally infeasible to find such an optimal partition.
Instead, we use a *top-down, greedy* approach:

- *Top-down:* start at the top
- *Greedy:* every split is the *best* split at this step, but may NOT be the best in the long term.

Search algorithm for decision trees

It is computationally infeasible to find such an optimal partition. Instead, we use a *top-down, greedy* approach:

- *Top-down:* start at the top
- *Greedy:* every split is the *best* split at this step, but may NOT be the best in the long term.

In other words, at every step when deciding which predictor to use for splitting (or call it a final region), the split is only constructed in a way such that the RSS at this split is minimized:

$$\sum_{i:\, x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:\, x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2,$$

Search algorithm for decision trees

It is computationally infeasible to find such an optimal partition. Instead, we use a *top-down, greedy* approach:

- *Top-down:* start at the top
- *Greedy:* every split is determined such that the *RSS at this split* is minimized.

This is also known as *recursive binary splitting.* At every step, we find a predictor and the cutpoint to split the data, until a user-defined stopping criteria has reached. E.g., all regions contain less than five observations.

Variable selection in decision tree: Tree pruning

A fully-grown tree could be specifically designed for the training set, and therefore not generalizable to test data. To prune this tree (i.e., only select a subset of predictor), we try to only keep the terminal nodes to minimize:

$$\sum_{m=1}^{|T|} \sum_{i:\ x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha|T|$$

- α is the tuning parameter,
- $|T|$ is the number of terminal nodes.
- R_m is the final region associates with the m^{th} terminal node.

Variable selection in decision tree: Tree pruning

A fully-grown tree could be specifically designed for the training set, and therefore not generalizable to test data. To prune this tree (i.e., only select a subset of predictor), we try to minimize:

$$\sum_{m=1}^{|T|} \sum_{i:\ x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

- α is the tuning parameter,
- $|T|$ is the number of terminal nodes.
- R_m is the final region associates with the m^{th} terminal node.

This is again a *loss+penalty* type of objective function, and the core of the minimization is equivalent to LASSO for regression problems.

Classification trees

Theoretically, if we think of a decision tree as dividing the **predictor space** to multiple regions $R_1, \dots, R_J$, each region has the exact same response value, then the regions should be divided such that

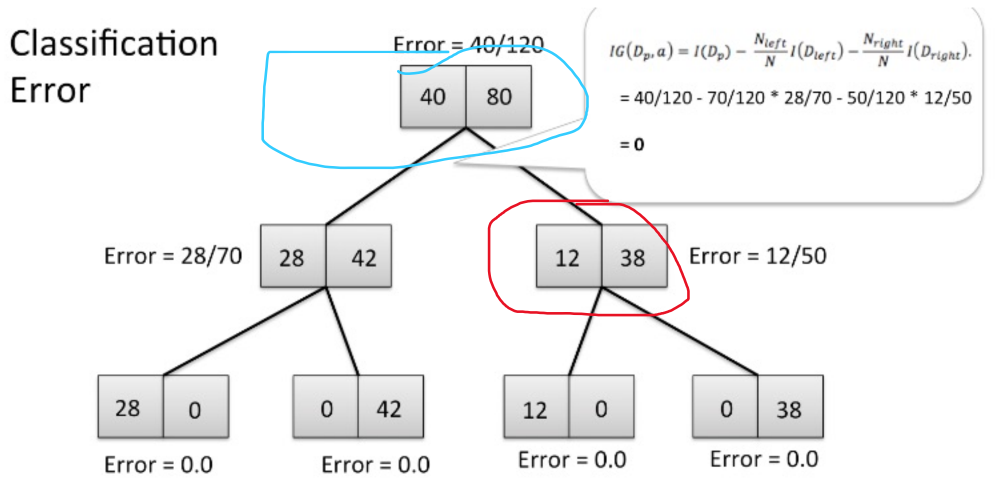
$$\text{minimize } \sum_{j=1}^{J} \sum_{i \in R_j} Error(y_i - \widehat{y_{R_j}}),$$

For classification, the predicted response for each region is naturally the majority class in this region. And the error function naturally becomes

Error of region j = 1 – accuracy at region j

Alternative approaches to error rate

However, in practice, error rate is NOT a preferred measure in decision trees. Consider the following tree:



$$IG(D_p, a) = I(D_p) - \frac{N_{left}}{N} I(D_{left}) - \frac{N_{right}}{N} I(D_{right}).$$

= 40/120 - 70/120 * 28/70 - 50/120 * 12/50

= 0

If our goal is to minimize error at each split, then the split at the second layer won't happen because 12/(12+38) is higher than 40/(40+80). In other words, this split would cause an increasing error, and thus won't be chosen.

Alternative approaches to error rate

Because error rate is not sensitive to tree-growing, in practice, we replace error rate with one of the following two measures:

- Gini index: $G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$, where $\widehat{p_{mk}}$ is the proportion of class k at node m.

When would Gini index be small? Consider the following three nodes:

1) At this node, 50% of the sample units are in class 1, the other 50% are in class 0.
2) At this node, 10% are in class 1, the other 90% are in class 0.
3) At this node, 100% are in class 1, 0% are in class 0.

Alternative approaches to error rate

Because error rate is not sensitive to tree-growing, in practice, we replace error rate with one of the following two measures:

- Gini index: $G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$, where $\widehat{p_{mk}}$ is the proportion of class k at node m.

When would Gini index be small? Consider the following three nodes:

1) At this node, 50% of the sample units are in class 1, the other 50% are in class 0.
2) At this node, 10% are in class 1, the other 90% are in class 0.
3) At this node, 100% are in class 1, 0% are in class 0.

Alternative approaches to error rate

Because error rate is not sensitive to tree-growing, in practice, we replace error rate with one of the following two measures:

- Gini index: $G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$, where $\widehat{p_{mk}}$ is the proportion of class k at node m. To minimize Gini index, we are forcing each node has a strongly *dominating* class k. Gini index is thus considered as a measure of *impurity.*

Alternative approaches to error rate

Because error rate is not sensitive to tree-growing, in practice, we replace error rate with one of the following two measures:

- Gini index: $G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$, where $\widehat{p_{mk}}$ is the proportion of class k at node m. To minimize Gini index, we are forcing each node has a strongly *dominating* class k. Gini index is thus considered as a measure of *impurity.*
- Entropy: $D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}.$

When would Entropy be small? Consider the following three nodes:

1) At this node, 50% of the sample units are in class 1, the other 50% are in class 0.
2) At this node, 10% are in class 1, the other 90% are in class 0.
3) At this node, 100% are in class 1, 0% are in class 0.

Alternative approaches to error rate

Because error rate is not sensitive to tree-growing, in practice, we replace error rate with one of the following two measures:

- Gini index: $G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$, where $\widehat{p_{mk}}$ is the proportion of class k at node m. To minimize Gini index, we are forcing each node has a strongly *dominating* class k. Gini index is thus considered as a measure of *impurity.*
- Entropy: $D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}.$

When would Entropy be small? Consider the following three nodes:

1) At this node, 50% of the sample units are in class 1, the other 50% are in class 0.
2) At this node, 10% are in class 1, the other 90% are in class 0.
3) At this node, 100% are in class 1, 0% are in class 0.

Alternative approaches to error rate

Because error rate is not sensitive to tree-growing, in practice, we replace error rate with one of the following two measures:

- Gini index: $G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$, where $\widehat{p_{mk}}$ is the proportion of class k at node m. To minimize Gini index, we are forcing each node has a strongly *dominating* class k. Gini index is thus considered as a measure of *impurity.*
- Entropy: $D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$. Numerically similar to Gini index.

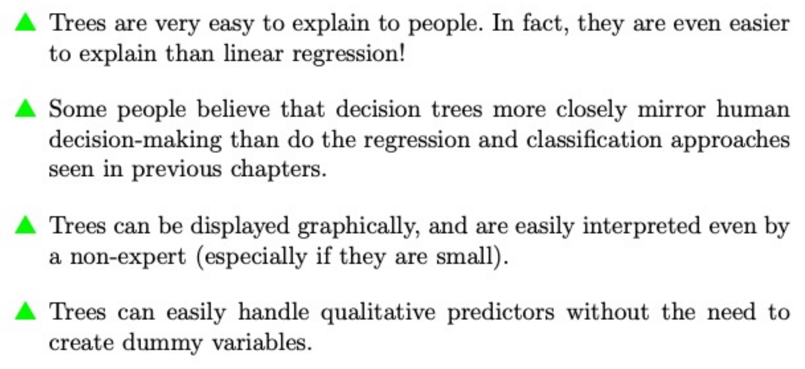
Summary of decision trees

To grow a decision tree:

1) At each step, we pick a predictor and a cutoff point to split the dataset, such that the *RSS/Gini Index/Entropy* resulted from this split is minimized.
2) We keep splitting until some user-defined stopping rule is reached.
3) After we fully grow a tree, we prune it such that only the terminal nodes that help minimize

$$\text{RSS/Gini/Entropy} + \alpha|T|$$

Pros and cons of decision trees

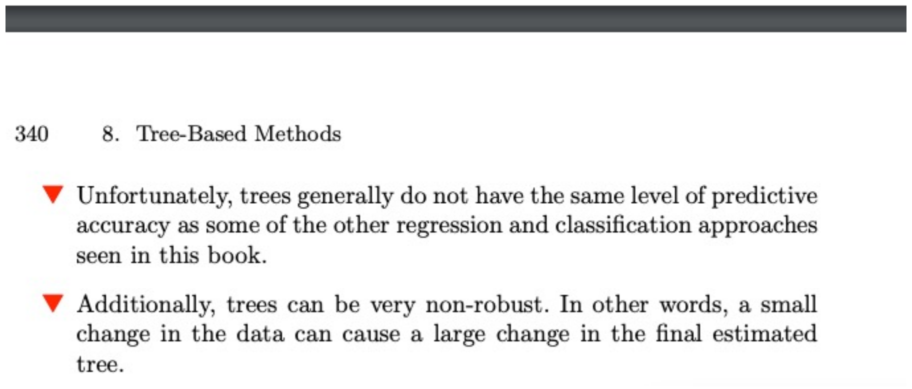


▲ Trees are very easy to explain to people. In fact, they are even easier to explain than linear regression!

▲ Some people believe that decision trees more closely mirror human decision-making than do the regression and classification approaches seen in previous chapters.

▲ Trees can be displayed graphically, and are easily interpreted even by a non-expert (especially if they are small).

▲ Trees can easily handle qualitative predictors without the need to create dummy variables.



▼ Unfortunately, trees generally do not have the same level of predictive accuracy as some of the other regression and classification approaches seen in this book.

▼ Additionally, trees can be very non-robust. In other words, a small change in the data can cause a large change in the final estimated tree.