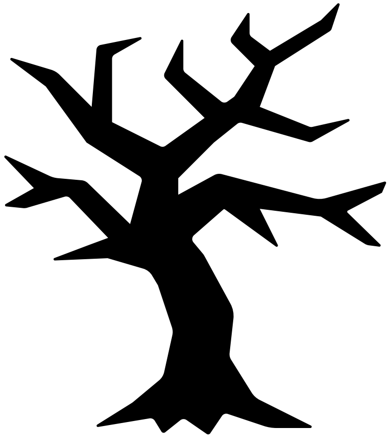
Chemical Engineering 4H03

Intro to Decision Trees (Classification)

Jake Nease
McMaster University



Objectives for this Topic

- This lecture is an **introduction** to the basics of decision trees used for binary classification
  – Note there are many more types (regression trees, classification trees without binary decisions)
- This content will **not be tested** on the final exam or A5
  – Here purely for your own interest and learning
- A brief outline
  – Introduction to the classification problem (with example)
  – Recursive partitioning
  – Pruning and testing
  – Random forest applications

A Brief History

- Decision trees are based on an intuitive application of human decision-making
  - First, a “major” decision is made
  - Subsequent decisions of gradually decreasing “importance” are made until a final verdict has been reached
  - Examples? Crossing the street? What courses to choose in final term of undergrad?
- Originally developed by Briemann *et al* (1984)
  - Known as “**C**lassification **a**nd **R**egression **T**rees” [**CART**]
- Consists of two major ideas:
  - Recursive partitioning (binary in our case)
  - Pruning and validation

Applications of Decision Trees

- Financial Sector
  - Assessing prospective growth opportunities
  - Using demographics to identify prospective clients
  - Using probabilistic outcomes to plan for future
- Engineering Sector
  - BP GasOIL system (decisions for routing and processing offshore oil instead of a hand-designed set of over 2,500 rules)
- Datamining and many other applications!
- Goofy applications
  - "Keep it or lose it" closet optimization guides
  - Do I make this Settlers of Catan trade?

Possible Types of Decision Trees

- Regression Trees
  - Attempt to predict an outcome (continuous or categorical) using continuous (or categorical) inputs
  - Each input category can include more than one outcome
  - Based on the minimization of **data entropy** as the tree descends

- **Classification Trees** (focus of this lesson)
  - Attempts to classify something based on continuous data
  - **Usually** results in a “yes or no” result
  - Attempts to minimize the **impurity** of a subspace of the data with regards to its classification
  - Attempts to minimize a metric known as the Gini Index

Classification Trees: The Idea

- The idea for a classification tree is fairly intuitive:

1. Find an independent variable and a critical value of that variable that **partitions** the data into two distinct subsets
   - We want these subsets to be as different from each other as possible as measured by the "purity" of dependent outcomes contained in that subset
   - Ideally, we want all of one outcome in one half, and all of another outcome in the other

2. Split the data into two subsets according to the previously chosen variable. Then, perform the same routine on BOTH of the subsets
   - This is why it is called "recursive partitioning"

A Motivating Example

- Example taken from Johnson and Wichern

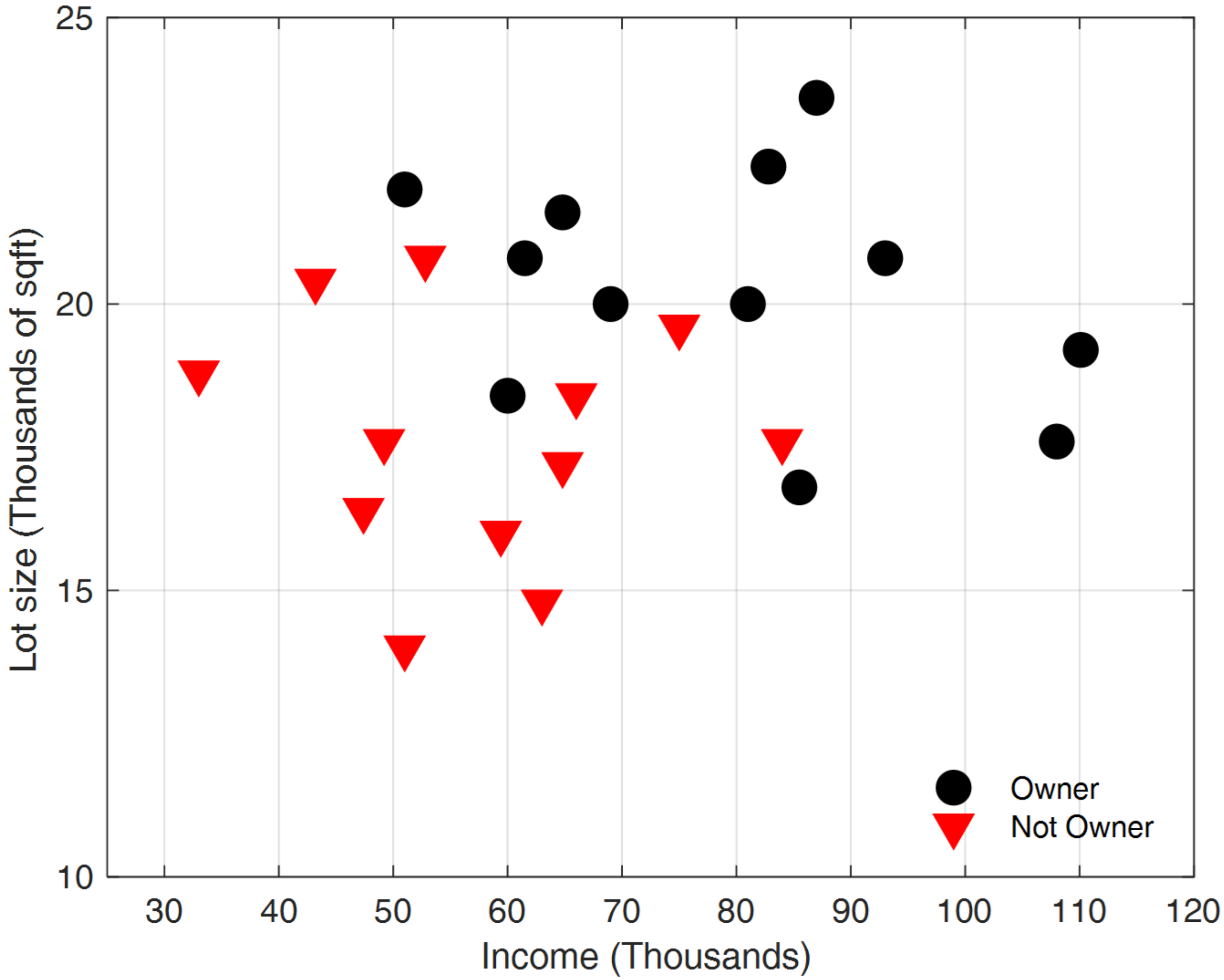
- Consider the set of data that represents 24 randomly canvased people regarding their ownership of a riding lawn mower

- We represent a riding mower manufacturer interested in knowing what types of families are likely to purchase a riding mower

Sample	Income (000s)	Lot Size (000s sqft)	Owns? (Y/N)
1	60	18.4	Y
2	85.5	16.8	Y
3	64.8	21.6	Y
4	61.5	20.8	Y
5	87	23.6	Y
6	110.1	19.2	Y
7	108	17.6	Y
8	82.8	22.4	Y
9	69	20	Y
10	93	20.8	Y
11	51	22	Y
12	81	20	Y
13	75	19.6	N
14	52.8	20.8	N
15	64.8	17.2	N
16	43.2	20.4	N
17	84	17.6	N
18	49.2	17.6	N
19	59.4	16	N
20	66	18.4	N
21	47.4	16.4	N
22	33	18.8	N
23	51	14	N
24	63	14.8	N

A Motivating Example

- We can make a nice pretty plot of this data:
  - I'll calmly mention that this plot works for up to 3D, not more



Can we think of any other way to classify this data…???

Would this be supervised or unsupervised…???

Partitioning: The Idea

- Consider a data set:

$$\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_i, y_i), \cdots, (\boldsymbol{x}_N, y_N)\} \triangleq \{X, \boldsymbol{y}\}$$

- This set contains:
  - Independent variables $\boldsymbol{x}_i^T = [x_{i,1}, x_{i,2}, \cdots x_{i,K}]$, where K is (as usual) the number of columns in X
  - Dependent (categorical) variable $y_i = \{1,2, \cdots, C\}$, where C is the total number of possible categories of $\boldsymbol{y}$

- So, in our example:
  - $x_i^T = [\text{income, lot size}]$
  - $y_i = \{\text{Owner, Not Owner}\}$

- Note that the values in $\boldsymbol{x}_i$ are continuous (for this example)

Partitioning: The Idea

- Partitioning involves dividing the K-dimensional space in X into non-overlapping **rectangles**
  - Rectangle in 2D, "hyper rectangle" for anything >2D
  - We'll just always call it a rectangle because words
- Each rectangle should be as homogeneous as possible
  - It should contain as many points as possible belonging to a single class in $\boldsymbol{y}$
- Once a rectangle is formed, one of the subsequent sub-rectangles is divided further using the same logic
  - This is the "recursive" part
- Thus, we will make more and more pure (and smaller and smaller) rectangles as we go

Measuring Homogeneity

- At each level in the tree, we want the subsequent rectangular subspace to be as **homogeneous** as possible

- Measuring homogeneity depends on the data set
  - For regression trees, one might use **information gain** or **entropy loss** from the split node
  - Essentially, this is a fancy way to say that the **variance** in the data decreases for each rectangular subspace
  - It is our goal with a decision split to **reduce the entropy** as much as possible by performing a split

- For classification trees, we can use a hard-nosed measure about how "pure" a rectangular subspace is
  - This is known as the **Gini Purity**

Gini Index

- When applied to decision trees, the Gini Index $\mathcal{G}$ is a measure of how "pure" a subspace is based on the likelihood of randomly drawing two samples from a data set and having them be the **same**

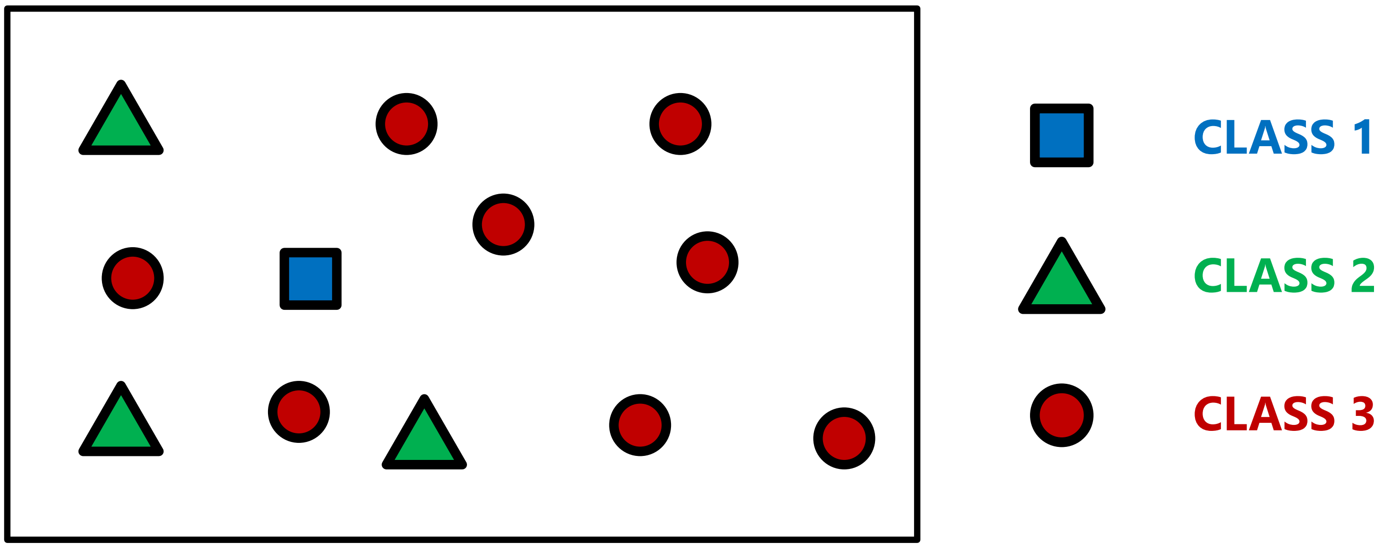
$$\mathcal{G} = \sum_{k=1}^{K} p_k^2 \qquad\qquad p_k = \frac{N_k}{\sum_{\ell=1}^{K} N_\ell}$$

- p_k is the **probability** that you will draw a random sample belonging to class $k \in K$ (just the ratio of points in k to all points)
- K is the total number of possible classes
- You might notice $\mathcal{G}$ has some nice properties:
  - $\mathcal{G} = 1$ if all samples in the subspace belong to the **same** class k
  - The lower $\mathcal{G}$, the more "impure" the subspace
  - The lowest possible $\mathcal{G} = 1 - \frac{K-1}{K}$ and represents a situation where all classes are **equally likely** to be chosen (perfectly mixed)

Gini Index Example

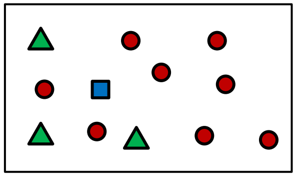
$$\mathcal{G} = \sum_{k=1}^{K} p_k^2$$

- Compute the Gini Index for the following subspace
  - Then consider: if the green points disappeared (but were still a viable class), what would $\mathcal{G}$ be?



- Questions:
  - How do these “classes” relate to our original example?
  - How does the “rectangle” relate to our original problem?

Gini Index Solution



- $\mathcal{G} = \sum_{k=1}^{3} p_k^2 = \left[\left(\frac{3}{12}\right)^2 + \left(\frac{1}{12}\right)^2 + \left(\frac{8}{12}\right)^2\right] = 0.514$ (not great)
- Without green points: $\mathcal{G} = 0.802$

Gini Impurity

- OK, so if $\mathcal{G}$ measures the "purity" of a subspace, then the "impurity" of a given subspace is nothing more than the complement to $\mathcal{G}$. We'll call this the Gini Impurity $\tilde{\mathcal{G}}$ and compute it as:

$$\tilde{\mathcal{G}} = 1 - \mathcal{G} \qquad [=] \qquad \tilde{\mathcal{G}} = 1 - \sum_{k=1}^{K} p_k^2$$

- Can you see where this is going?
  - That's right… We are going to attempt to split a given set of data into two rectangles, HOPING that the Gini Impurity of the resulting rectangles is considerably lower than what we started with
  - We will do this in a **Greedy-Algorithm** approach where we split each rectangle with this objective (without worrying about future splits)

Decision Tree Algorithm

- Given a set of training data X belonging to a rectangular space $\mathcal{R}$ with each $\boldsymbol{x}_i$ belonging to a given class y_i

1. For each dimension in X

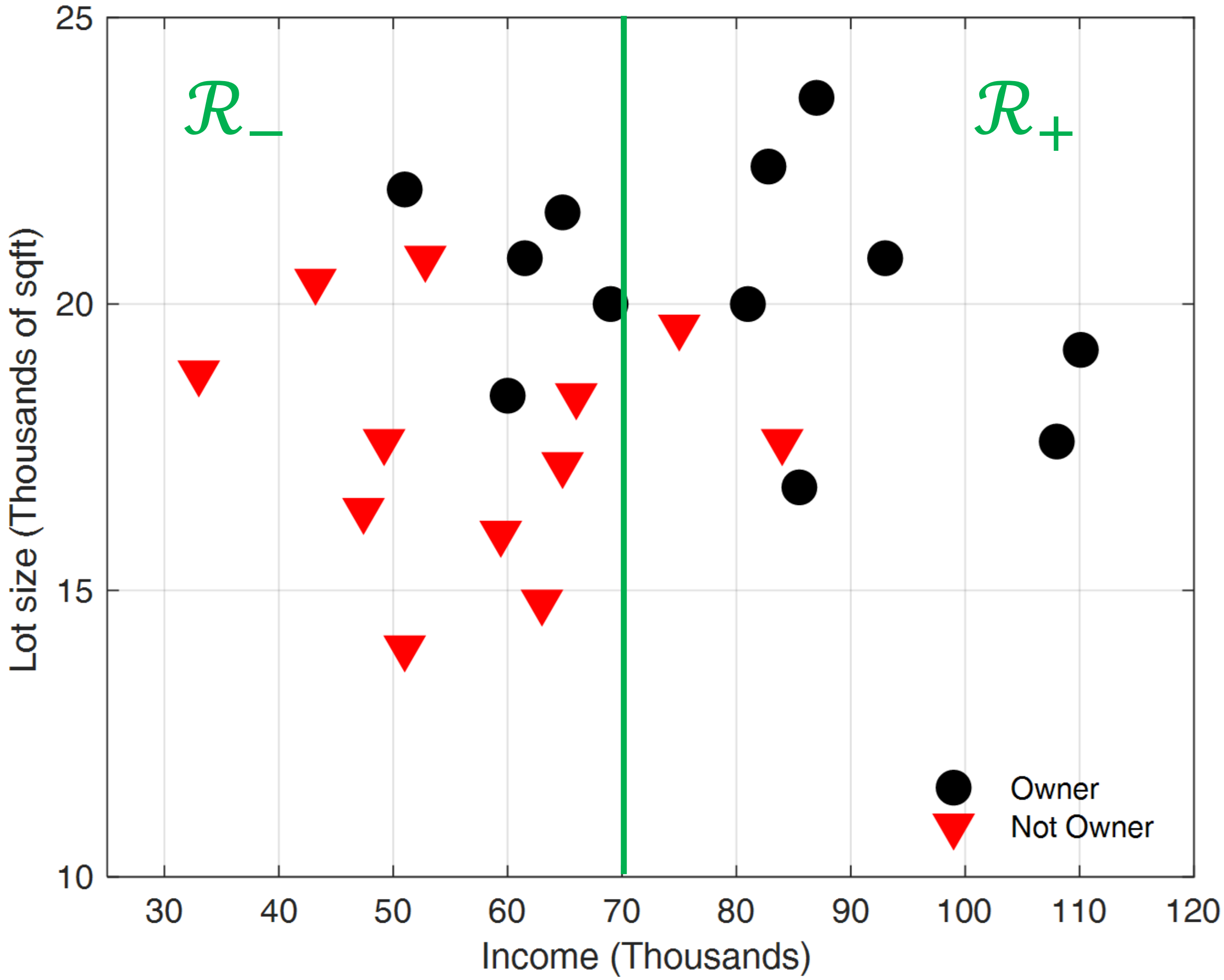
   Identify a threshold that, if used to split the data into two further rectangles, will minimize the resulting $\tilde{\mathcal{G}}$ classifying $\boldsymbol{y}_i$
2. Select the dimension and threshold that corresponds to the minimized **weighted** $\tilde{\mathcal{G}}_{\pm}$ amongst all candidates
3. Separate the training data belonging in $\mathcal{R}$ to the two new rectangles they belong to: $\mathcal{R}_-$ and $\mathcal{R}_+$
4. Recursively apply steps (1)-(3) to $\mathcal{R}_-$ and $\mathcal{R}_+$ until perfect homogeneity in all $\mathcal{R}$ is achieved

Decision Tree Algorithm Notes

- Since this is a recursive algorithm, we really only need to figure it out once
  - Code it as a function or something and spam the resulting subspaces into itself until all $\tilde{\mathcal{G}} = 0$
- We use the **weighted** impurity $\tilde{\mathcal{G}}_{\mp}$ to avoid moving a single point into its own subspace and saying "ooh look at how good a job I did!"
- Also note the nomenclature $\mathcal{R}_-$ and $\mathcal{R}_+$ is of my own invention and is not technically correct
  - Would need a name for each sub-rectangle, but if we are doing it recursively we only really need to worry about what is in front of us
- The decision tree will DEFINITELY be over-fit
  - We'll worry about this later

Back To Our Example

- Suppose we had a proposed split on INCOME at 70k
  - What would the resulting $\tilde{\mathcal{G}}_{\pm}$ be?



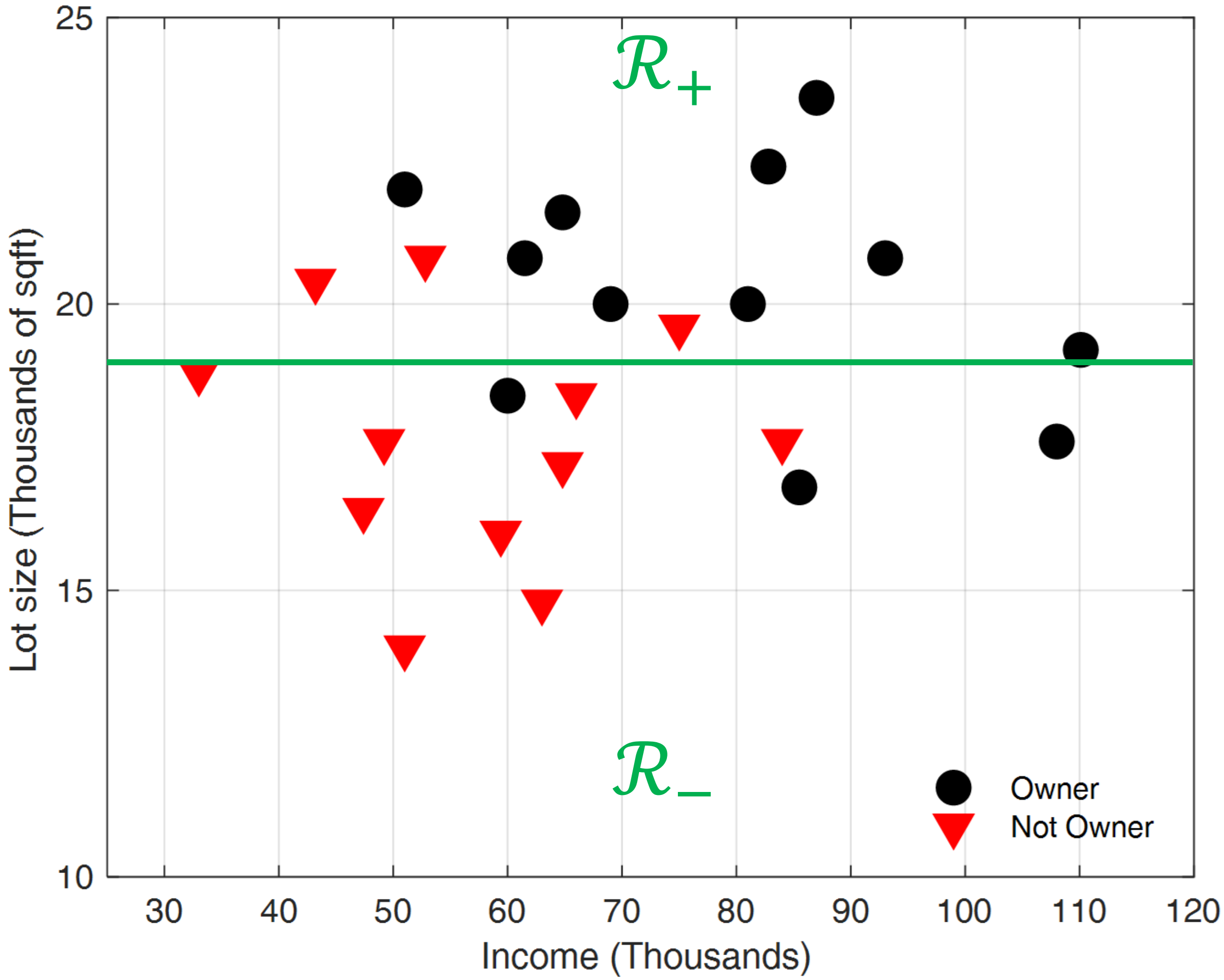
Example Solution

- Suppose we had a proposed split on INCOME at 70k
  - What would the resulting $\tilde{\mathcal{G}}_{\pm}$ be?

$$\tilde{\mathcal{G}}_{\pm} = 0.4074$$

One More Time

- Suppose we had a proposed split on LOT at 19k
  - What would the resulting $\tilde{\mathcal{G}}_{\pm}$ be?



Example Solution

- Suppose we had a proposed split on INCOME at 70k
  - What would the resulting $\tilde{\mathcal{G}}_{\pm}$ be?

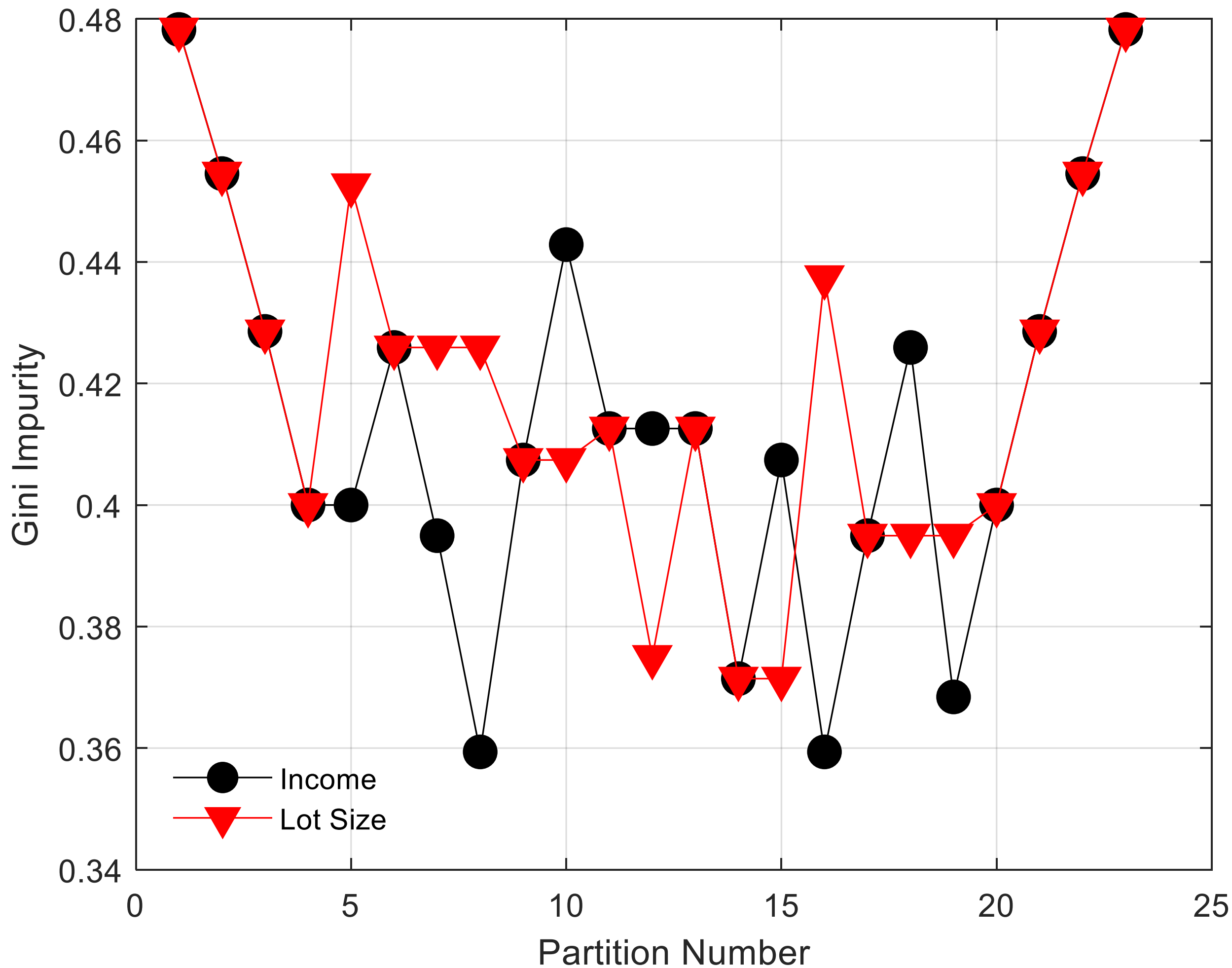
$$\tilde{\mathcal{G}}_{\pm} = 0.3750$$

So Now What?

- Well, based on those results, the logical thing to do would be to split on LOT = 19k (lower $\tilde{\mathcal{G}}_{\pm}$)
- But is that optimal?
  - Nope. We need to be a little more thorough
- Choosing the dimension and threshold depends on the data type of the independent variable
  - If DISCRETE, simply test each possible direction (eg, in our data set if we had a dimension $k = \{\mathrm{OWNS}, \mathrm{RENTS}\}$ house)
  - If CONTINUOUS, we go through the data one point at a time and **take its average with the next highest**, try splitting there, record the $\tilde{\mathcal{G}}_{\pm}$, and proceed
  - In either case, we track the current global-best $\tilde{\mathcal{G}}_{\pm}$ and once we have tried everything, we commit to that!

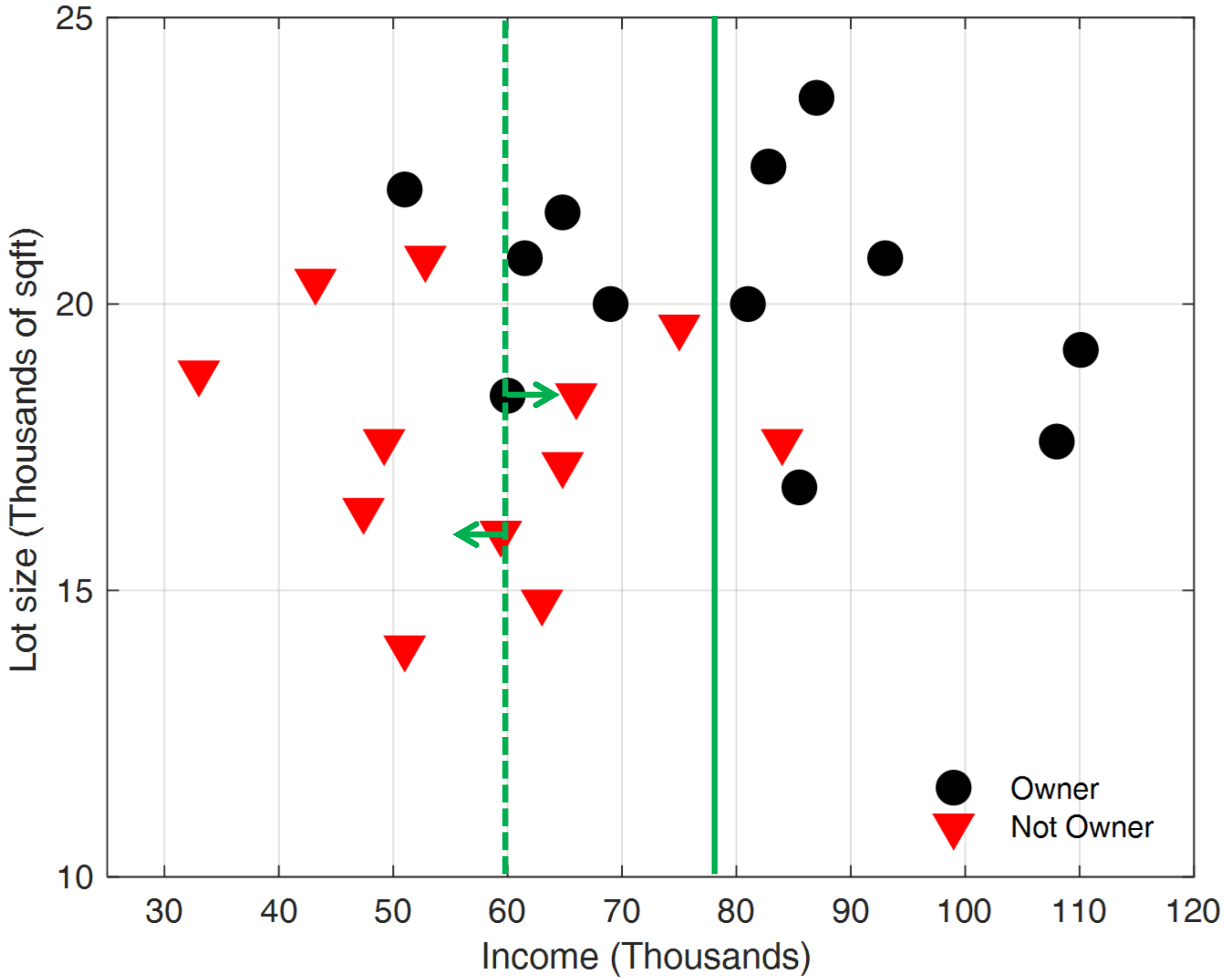
Visualizing $\tilde{\mathcal{G}}_{\pm}$ for our Data

- It seems the best split is on INCOME at partition 8 (59.7k) or partition 16 (78k). Both Equally Good!



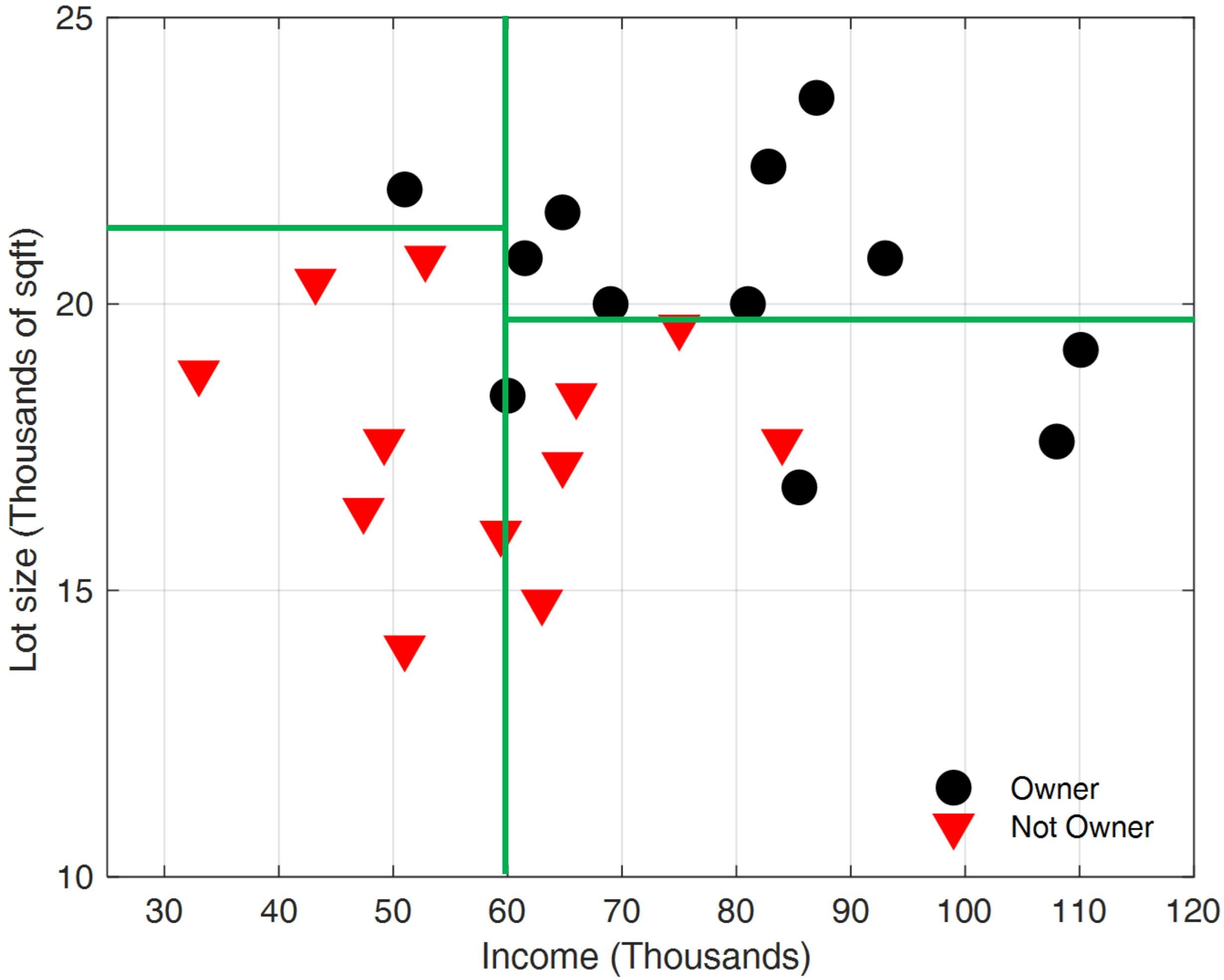
Back To Our Example

- We can visually verify either is a good split in this case



And So On…

- We then take each of the two subspaces and if any impurities exist, we recursively split again!



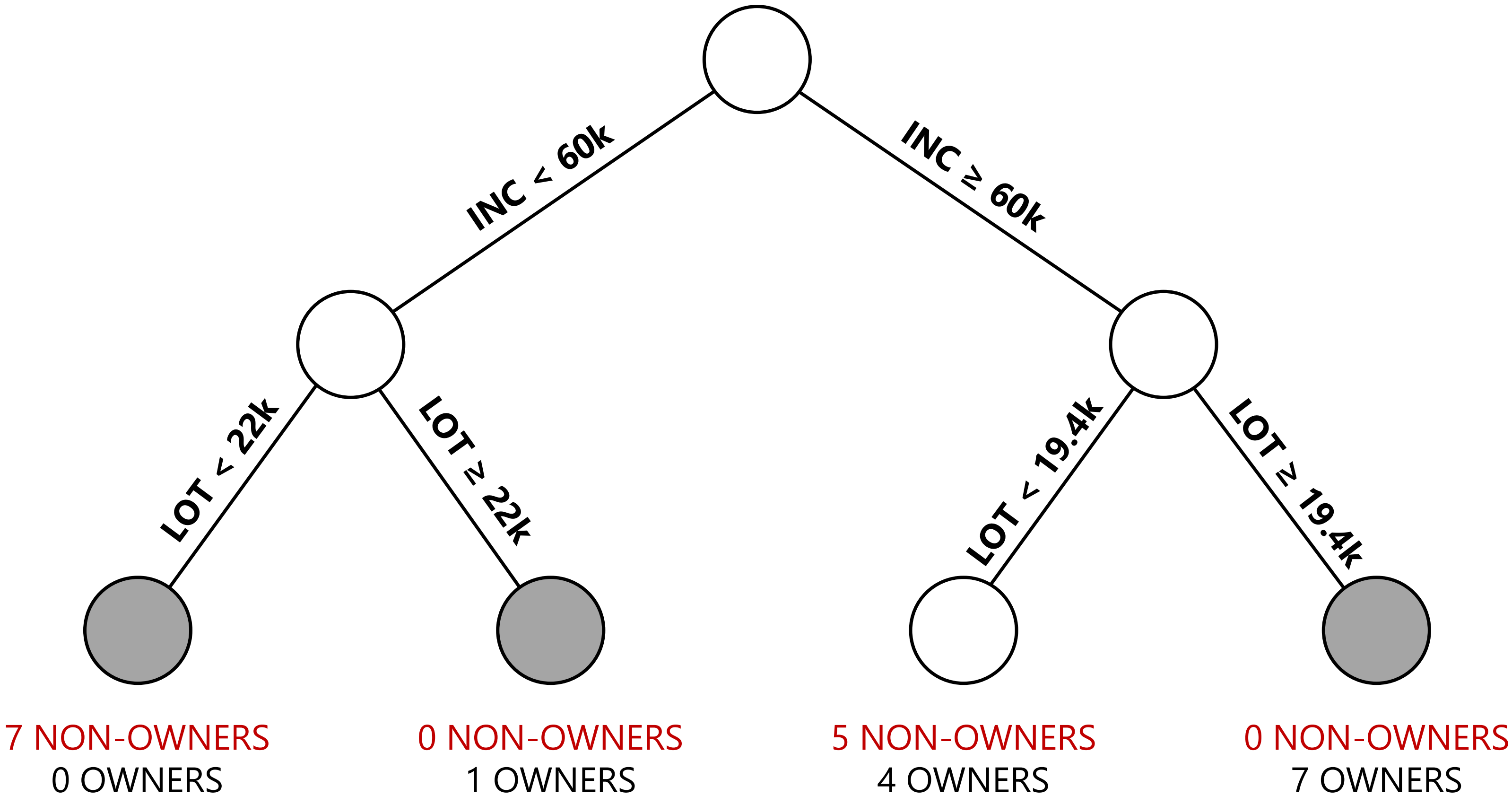
OK, I know What You're Thinking

WHERE THE BLOODY H*...**

IS THE "TREE" IN ALL THIS??

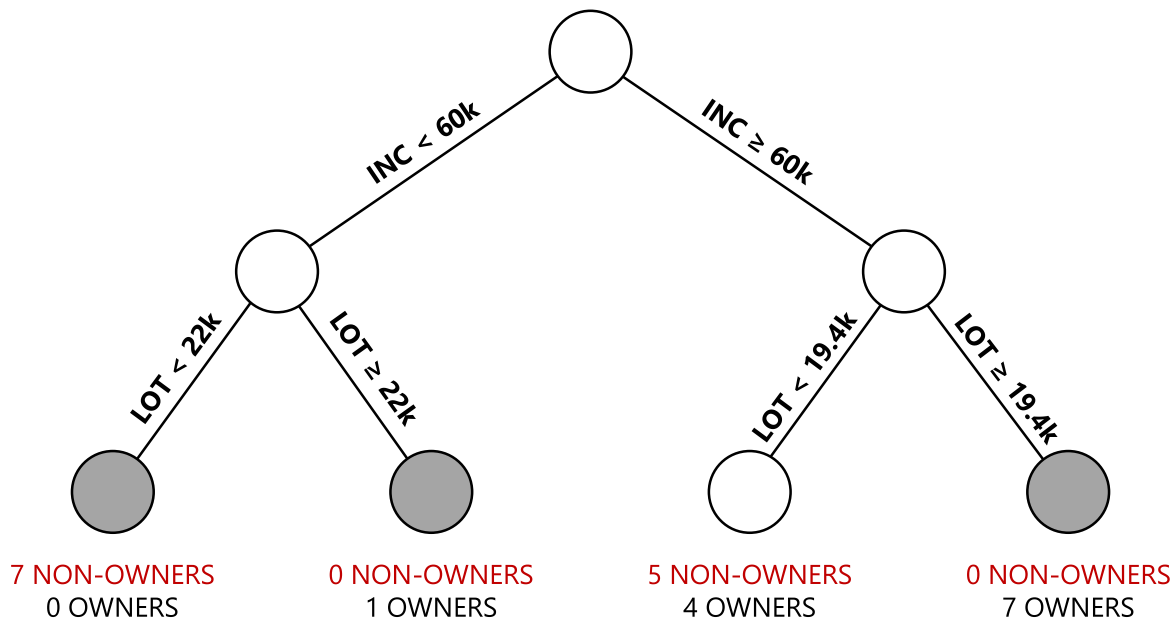
The “Decision Tree”

- The TREE is a graphical representation of how the search space is sub-divided into its rectangles!



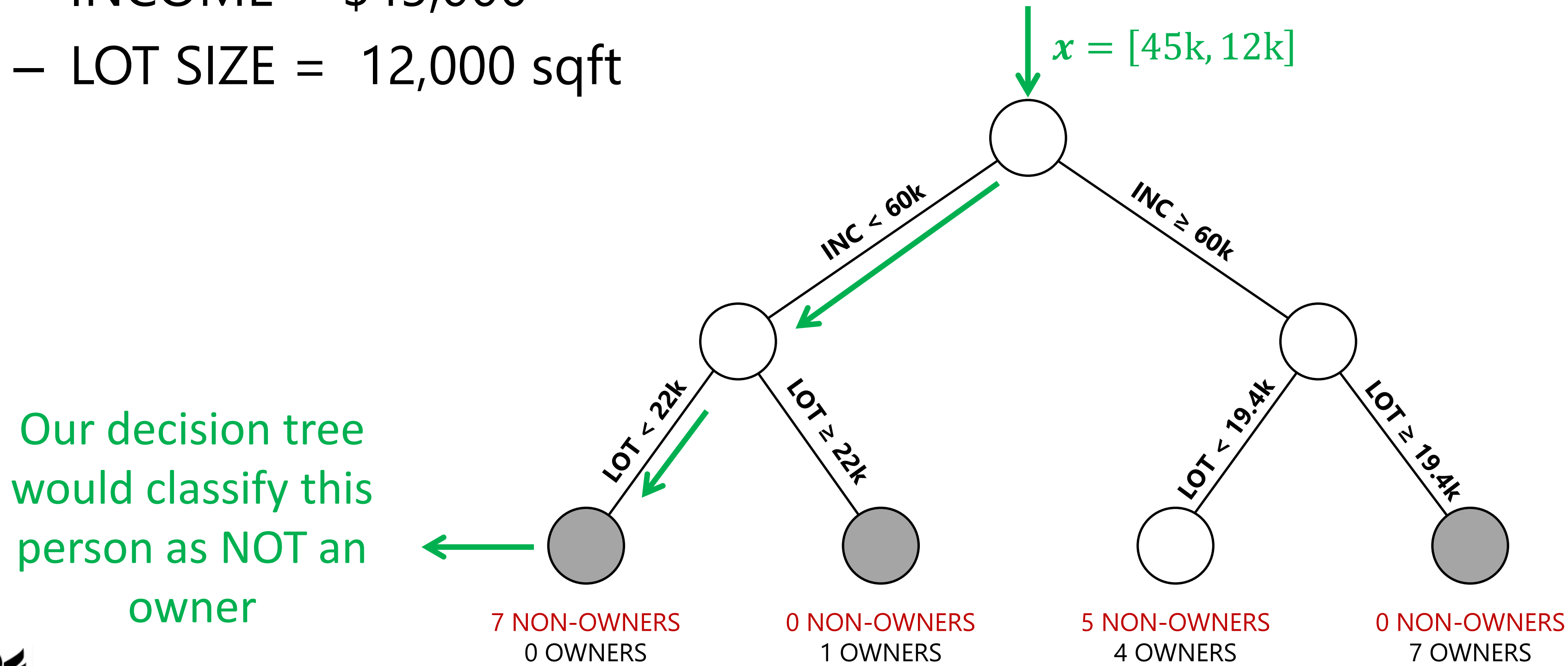
The "Decision Tree"

- In our case, we would consider the grey-shaded nodes as **COMPLETE**
  – In decision tree jargon, they are called **LEAVES** (cute, right?)
- The white lowermost node is incomplete (contains *entropy*, as they say… We just call it variance)
  – In other words, that node is impure according to $\mathcal{G}$
  – We need to **BRANCH** on that node (still cute!)



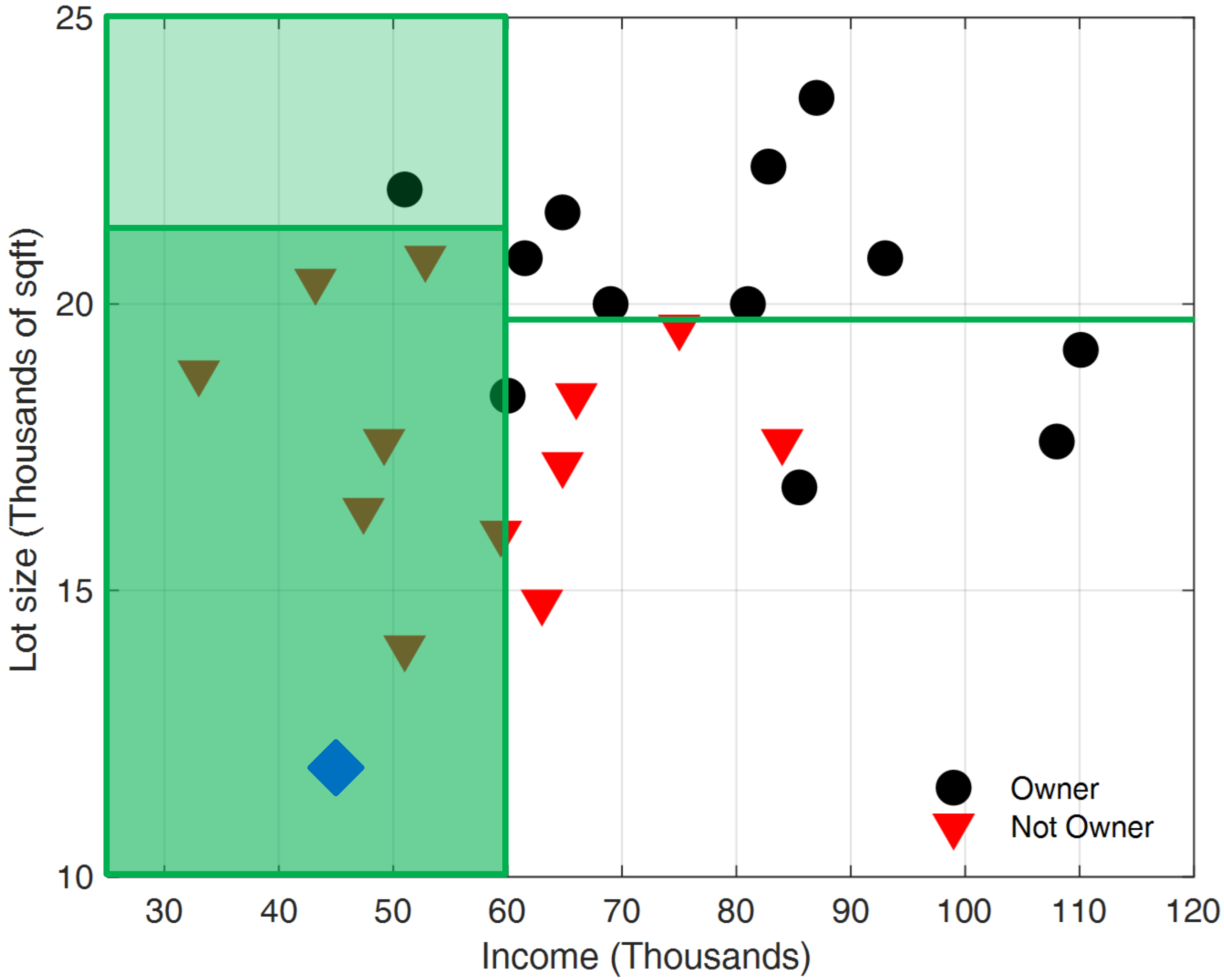
Passing Data Into the Tree

- Once all branches become leaves, any new data follows the same set of decisions until it is classified according to that leaf
- EXAMPLE: Consider a customer:
  - INCOME = $45,000
  - LOT SIZE = 12,000 sqft



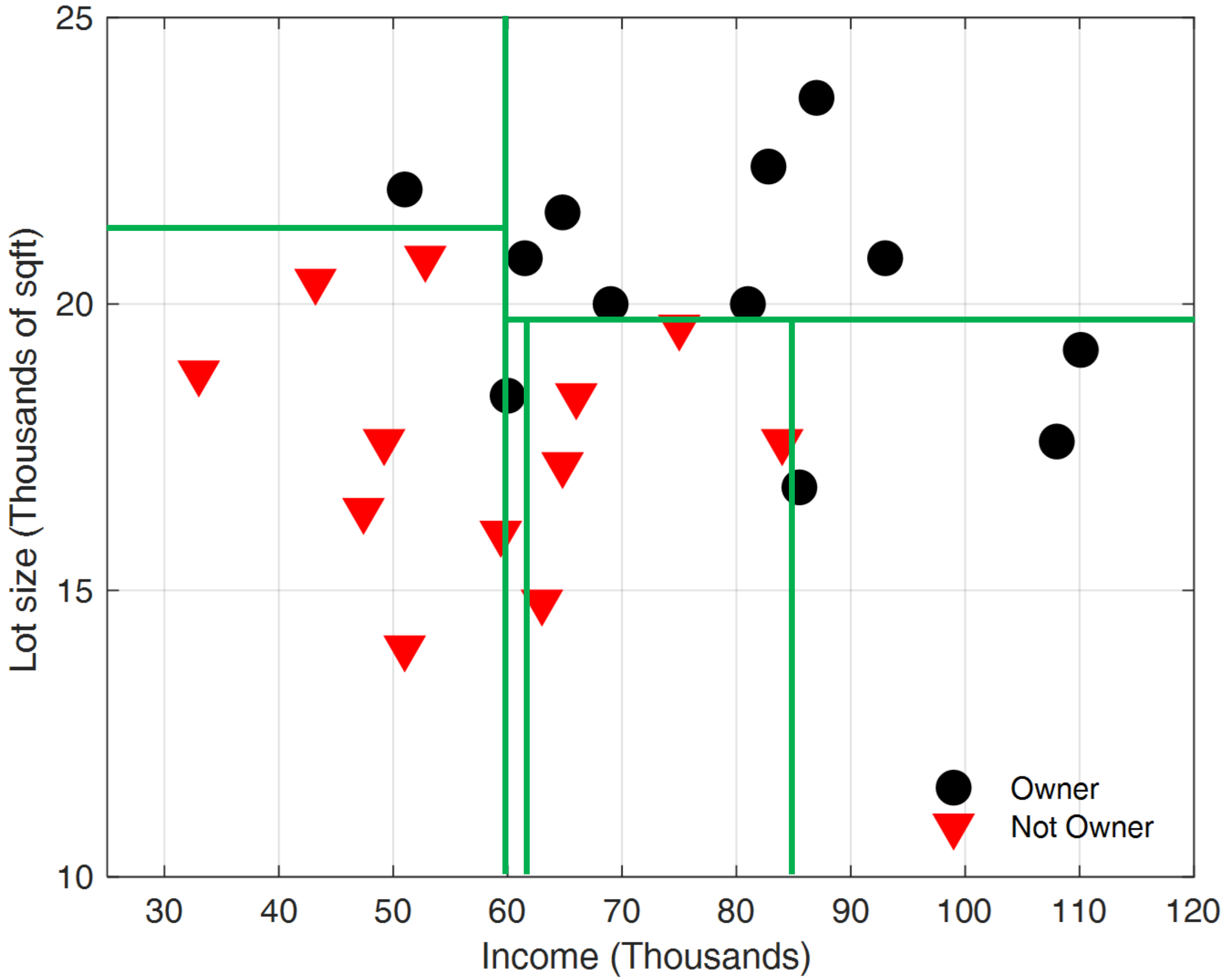
In Our Rectangular Subspaces

- First we segregate into left region, then lower region!



So Are we Done with the Tree?

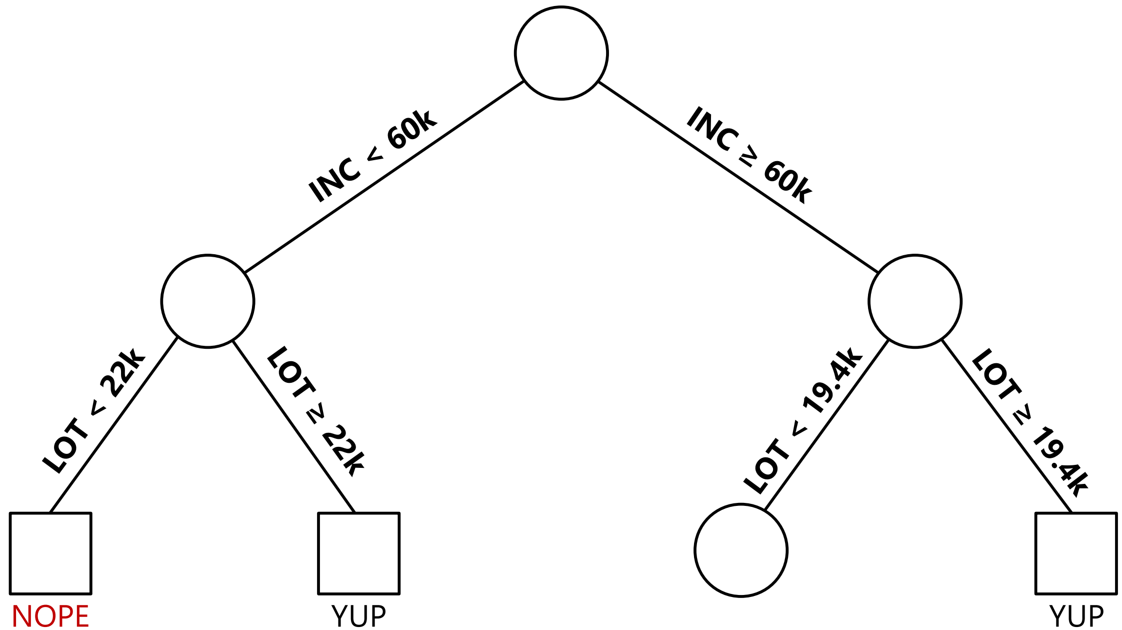
- Nope. We need to keep going until all nodes are pure!
  - Here is the “completed” subspaces



Workshop

- **Draw** the completed decision tree for the completed subspaces

- **Special notes**:
  - A square node is the pictographic representation of a “leaf”
  - Fully completed (blossomed?) trees have pure leaf classification, hence the conclusions “nope” and “yup”
  - REMEMBER that you can keep splitting on the same variable (we used INC then LOT in that order only by coincidence)



Pruning

- Decision trees will always be overfit
  - If allowed to grow to completion, there will be zero error in the training set's predictions
  - Obviously, a testing set may say otherwise

- Strategy (in CART) is to fit the entire tree first, then employ some sort of cost-based **pruning** strategy that cuts off branches, replacing them with leaves
  - The terminal solution to the pruned tree will be the majority vote of the results contained in the leaf
  - Uses a cost-complexity objective function that depends on the user's preference

Cost Complexity

- CART computes a cost complexity objective like:

$$\min \phi = \sum \#\text{misclassifications} + C(\#\text{leaves in tree})$$

- For a regression tree, something similar:

$$\min \phi = \sum_{x_i} (\hat{y}_i - y_i)^2 + C(\#\text{leaves in tree})$$

- In any case:
  - The cost is computed for the given tree
  - The sub-tree with lowest **information gain** is removed
  - If ϕ decreases, that subtree is eliminated and replaced with a leaf
  - Repeat until no more sub-trees eliminated

Some Other Splitting Functions

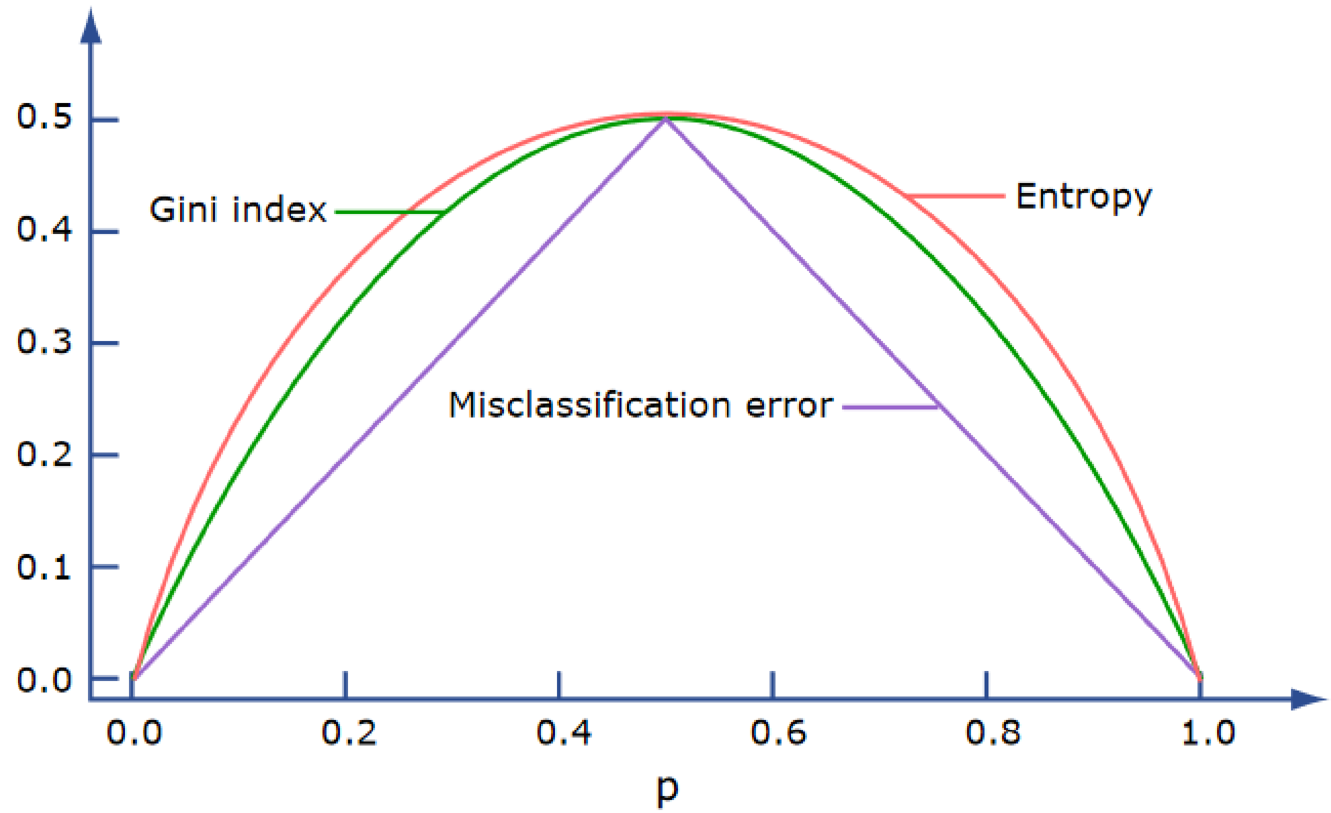
- Instead of the Gini Index, it is possible to split a classifier based on other metrics
  - Provided here just as a reference

- For example, you may choose to split based on **ENTROPY**

$$H(y_i) = -\sum_{k=1}^{C} P(y_i = Y_k) \log_2\big(P(y_i = Y_k)\big)$$

- In this case, $H(y)$ is the entropy of the outcome for any y_i based on the probability it belongs to class Y_k for all $k = 1 \ldots C$ possible classes
  - Entropy is maximized when all probabilities are equal
  - Entropy is minimized when one probability is a certainty
  - Tree aims to minimize remaining entropy in data after each split!

Some Other Splitting Functions



Node impurity measures for two-class classification, as a function of the proportion p in class 2. Cross-entropy has been scaled to pass through (0.5, 0.5).

Image by MIT OpenCourseWare, adapted from Hastie et al., *The Elements of Statistical Learning*, Springer, 2009.

Brief Intro to Random Forests

Sometimes 500 is Better than 1

Primary Concept Of Random Forests

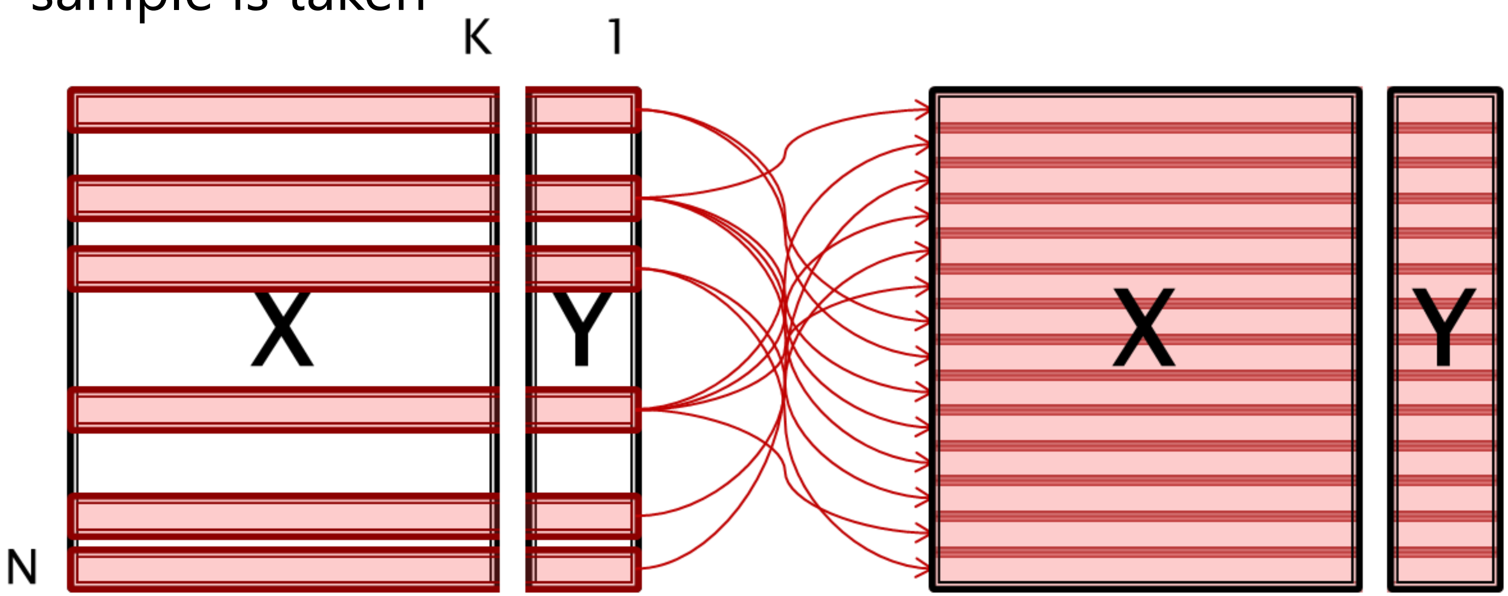
- Instead of fitting one large decision tree, numerous smaller trees are trained using **random** samples of the data and examining only **random** columns
- Idea: pass a new point $\boldsymbol{x}$ **into** an *ensemble* of B trees $\mathcal{T}$

$$\mathcal{T} = \{T_1, T_2, \cdots, T_B\}$$

- Importantly, an individual tree T_b evaluates $\boldsymbol{x}$ based on only the columns with which it was trained
  - Resulting classification for T_b is recorded
- Votes are tabulated after ALL trees have seen the point $\boldsymbol{x}$ and results in one of two outcomes:
  - Strict classification (majority vote)
  - Probability of outcome (average of votes)

Training a Random Forest

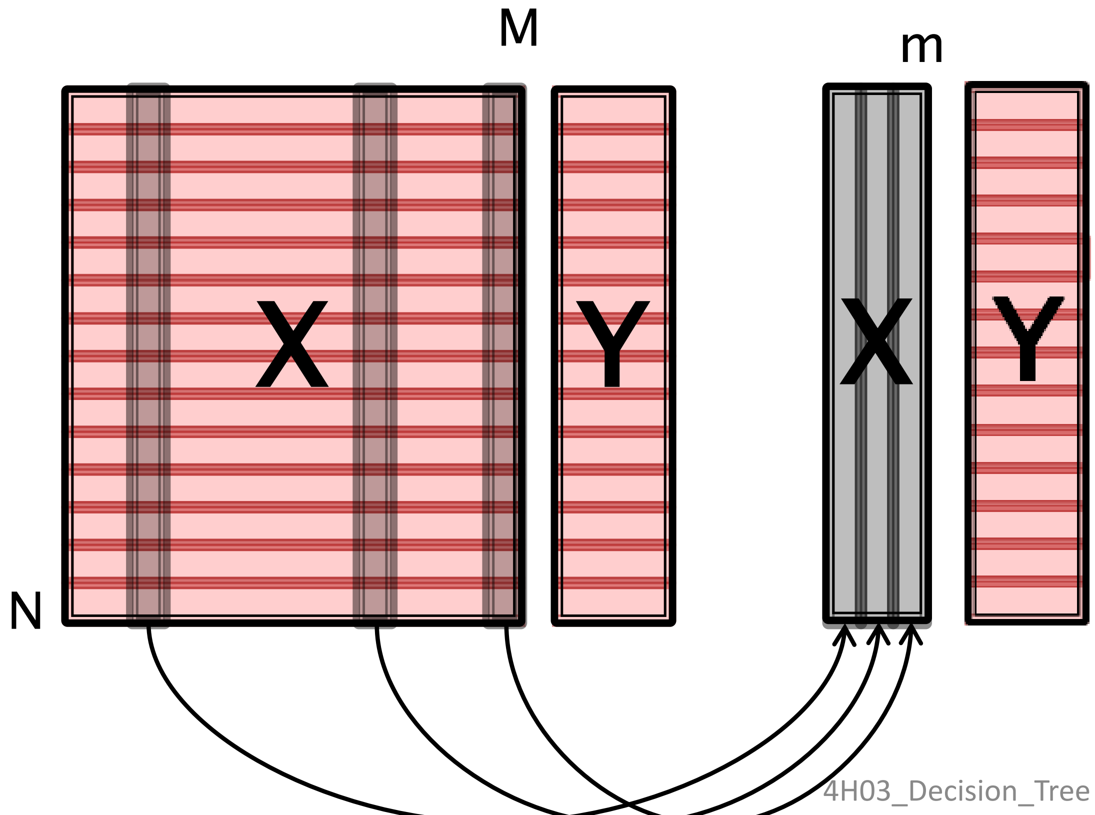
- Given training block **X** and corresponding outcomes $\boldsymbol{y}$
- Draw a so-called **Bootstrap Sample** from **X**
  - Bootstrapping is equivalent to sampling **with** replacement
  - Suggestion: remove $\approx \frac{1}{3}$ of the data from **X** before each sample is taken



Training a Random Forest

- Given training block **X** and corresponding outcomes $\boldsymbol{y}$

- Extract m columns out of M in the bootstrap sample
  - m is typically chosen to be 2-4

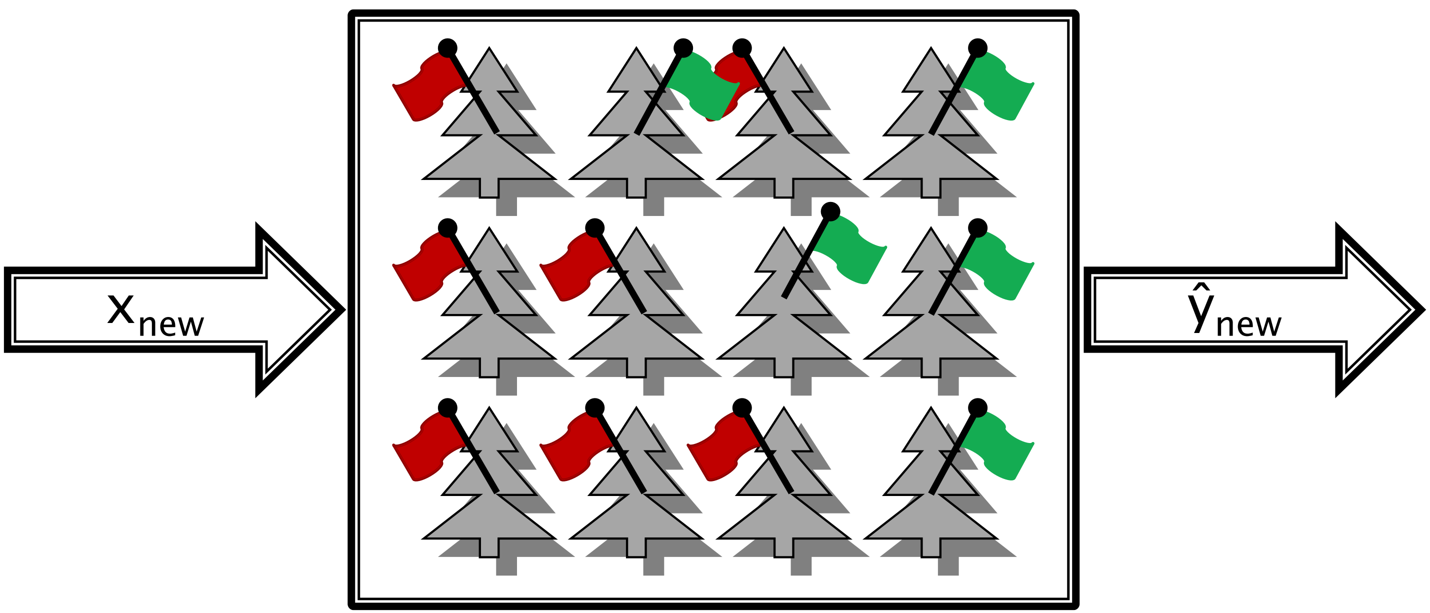


Training a Random Forest

- Given training block $\mathbf{X}$ and corresponding outcomes $\boldsymbol{y}$
- Grow a FULL decision tree on those m columns using the bootstrapped samples
  - Do NOT prune the tree
- Repeat this B times for the number of trees in the "forest"
- Can choose B manually or via cross-validation
  - Since 1/3 of data was hidden from each tree, cross-validation is possible
  - Metrics such as mean-squared error (regression trees) or misclassification rate can be measured after tree b

Using a Random Forest

- A new row x is passed into the forest and processed by each decision tree
  - Each tree will classify x based only on the columns it has access to
  - Votes counted amongst all trees



Advantages of Random Forests

- Computation time is quite efficient
  - Growing many small trees without pruning is much faster than growing a large tree with pruning when K is very large
- Over-fitting is not a problem
  - *Strong Law of Large Numbers* (see this reference)
- Classification accuracy is very high
- Robust against non-relevant descriptors
- Can be used on X blocks where K > N
- Can be used on highly nonlinear data sets
  - Good when an appropriate pre-processing nonlinear transformation is not known

Predicting Loan Delinquency

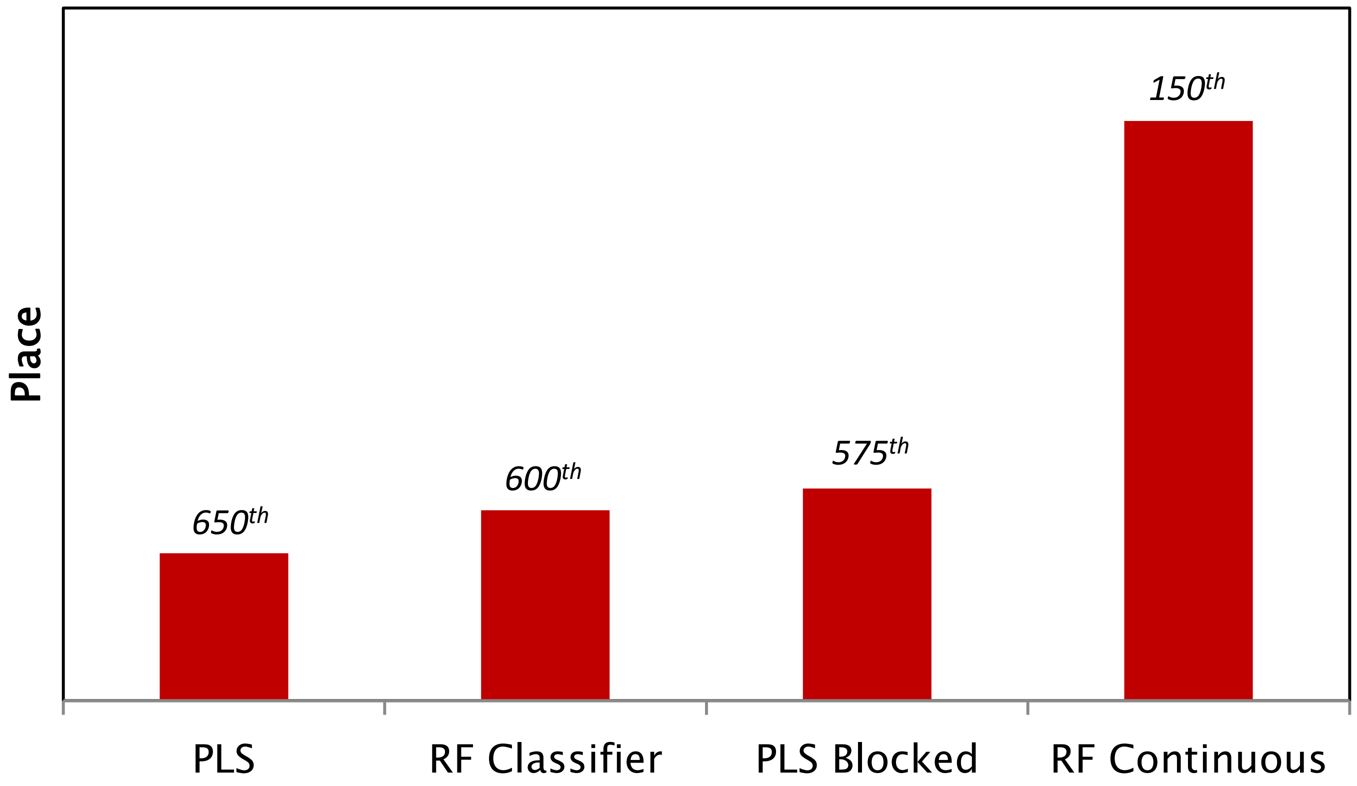
- Case study from Kaggle
- Banking data used to predict loan delinquency rates based on financial information
  - Credit cards
  - Mortgages
  - Loans
  - Previous delinquencies
  - Credit scores

- Bank wishes to predict whether or not a debtor has a legitimate chance of delinquency

- Data is 150,000 rows and 10 columns

- For comparison, PLS was also fit

Results were Strong



One Small Point

- For this study, it was found that continuous classifier trees were more successful than binary classifiers

- Did not account for missing data
  - PCA to impute?