CS167: Machine Learning

Decision Tree Implementation

Decision Tree (classifier/regressor)
Train/test split
Evaluation metrics

Wednesday, March 4th, 2026

Drake UNIVERSITY

Announcements

- Heads up that Quiz #1
  - Will take place on Wednesday (03/11) during class time
  - All the topics up to this week
  - Types of questions:
    - MCQ
    - True/False
    - Fill in the blanks (may or may not require calculations)

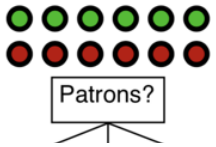
Announcements

- Notebook #3 Cross Validation with kNN and Vehicle Fuel Efficiency
  - Due tonight: 03/04 by 11:59pm
  - To submit, download the `ipynb` file from Colab

Review: Summarizing all three steps!

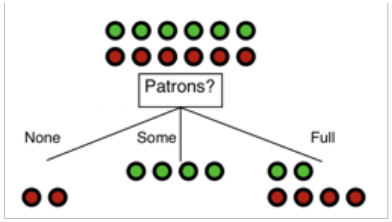
- **Step 1:** Calculate the entropy of the distribution of the classes before the node you are testing. This is the **entropy before**

Entropy_before(Patron) = 1



- **Step 2:** Calculate the **expected entropy**
  - The weighted sum of the entropy of each split of the data

Expected_entropy(Patron) = 0.459



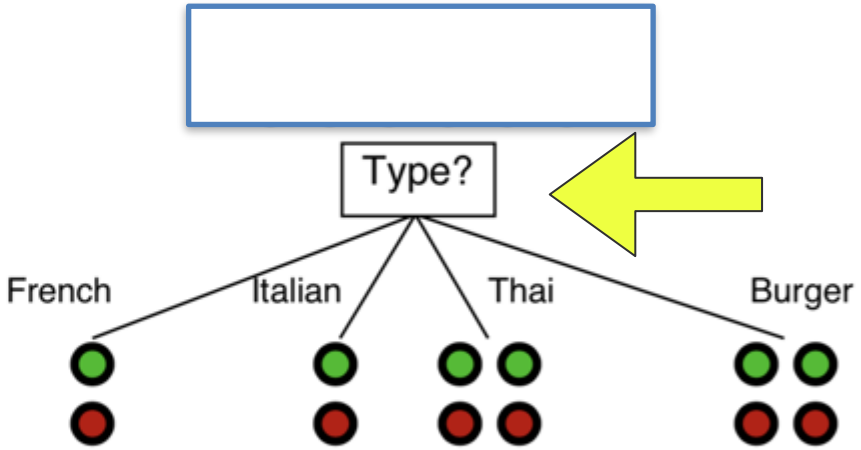
- **Step 3:** Find the difference between the **entropy before** and **expected entropy**
  - **Information Gain(Patron) = Entropy_before(Patron) - Expected_entropy(Patron)**

= 1 - 0.459

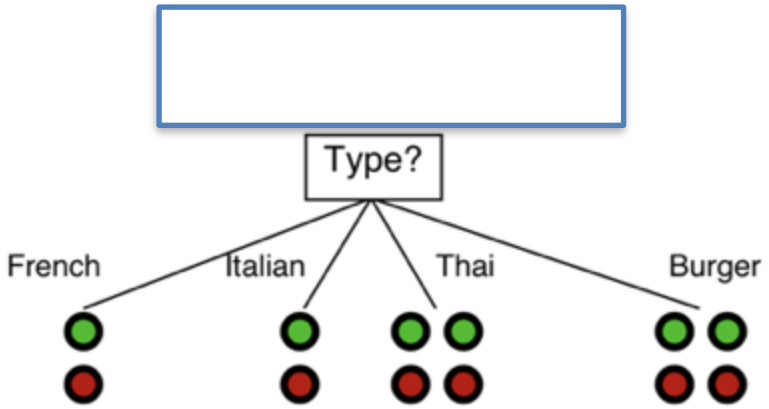
= 0.541

Review: Another Example: Information Gain

- Calculate entropy for feature *Type* as a candidate



Review: Another Example: Information Gain



Note that the expected entropy for the *Type* feature is

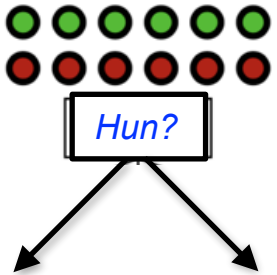
$$\frac{2}{12} \cdot Entropy\left(\left\langle \frac{1}{2}, \frac{1}{2} \right\rangle\right) + \frac{2}{12} \cdot Entropy\left(\left\langle \frac{1}{2}, \frac{1}{2} \right\rangle\right)$$
$$+ \frac{4}{12} \cdot Entropy\left(\left\langle \frac{2}{4}, \frac{2}{4} \right\rangle\right) + \frac{4}{12} \cdot Entropy\left(\left\langle \frac{2}{4}, \frac{2}{4} \right\rangle\right)$$
$$= \frac{2}{12} \cdot 1 + \frac{2}{12} \cdot 1 + \frac{4}{12} \cdot 1 + \frac{4}{12} \cdot 1 = 1$$

So,

$$Gain(Type) = 1 - 1 = 0$$

Decision Tree: Exercise Information Gain

- Calculate the Information Gain for *Hun*:

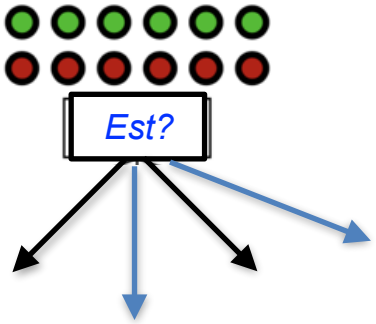


Target feature

Ex	*Alt*	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	Wait
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

Decision Tree: Exercise Information Gain

- Calculate the Information Gain for *Hun*:

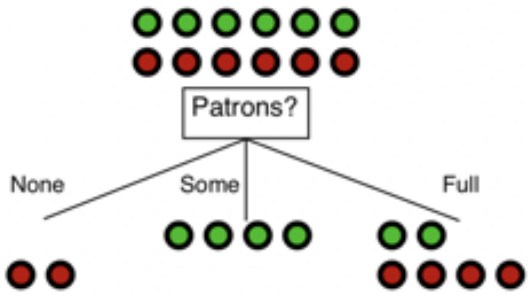


Target feature

Ex	*Alt*	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	Wait
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

Decision Tree: Exercise Information Gain

- Information Gain results
  - Gain(Alt) = 1 - 1 = 0
  - Gain (Bar) = 1 - 1 = 0
  - Gain(Fri) = 1 - 0.979 = 0.021
  - Gain(Hun) = 1 - 0.804 = 0.196
  - **Gain(Patrons) = 1 - 0.459 = 0.541**
  - Gain(Price) = 1 - 0.804 = 0.196
  - Gain(Rain) = 1 - 1 = 0
  - Gain(Res) = 1 - 0.979 = 0.021
  - Gain(Type) = 1 - 1 = 0
  - Gain(Est) = 1 - 0.792 = 0.208



Decision Tree: What to do with numeric features?

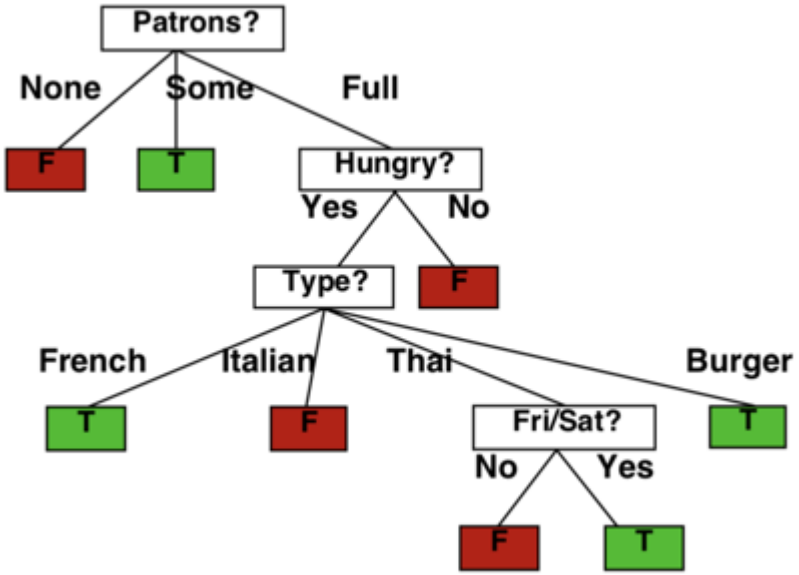
What do we do if we have numeric (even continuous-valued) features like age from the titanic dataset or petal length from the iris dataset?

Idea: Decision Tree thresholds: if age > 70

Unfortunate annoying thing: Even though decision tree algorithms work well with categorical data, the Python library we will work with still wants all predictor features converted to a number, so we will have to work with numbers no matter what.

Decision Tree Size Discussion

Decision tree learned from the 12 examples:



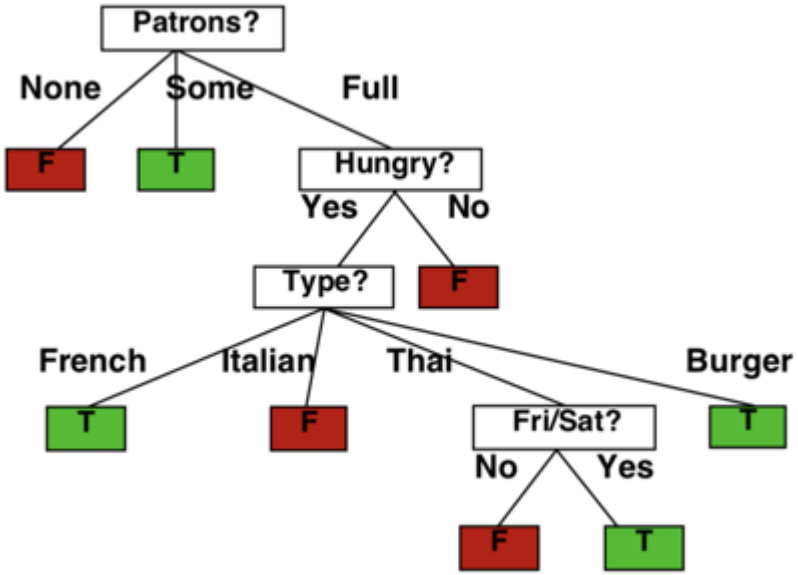
Decision Tree Size Discussion

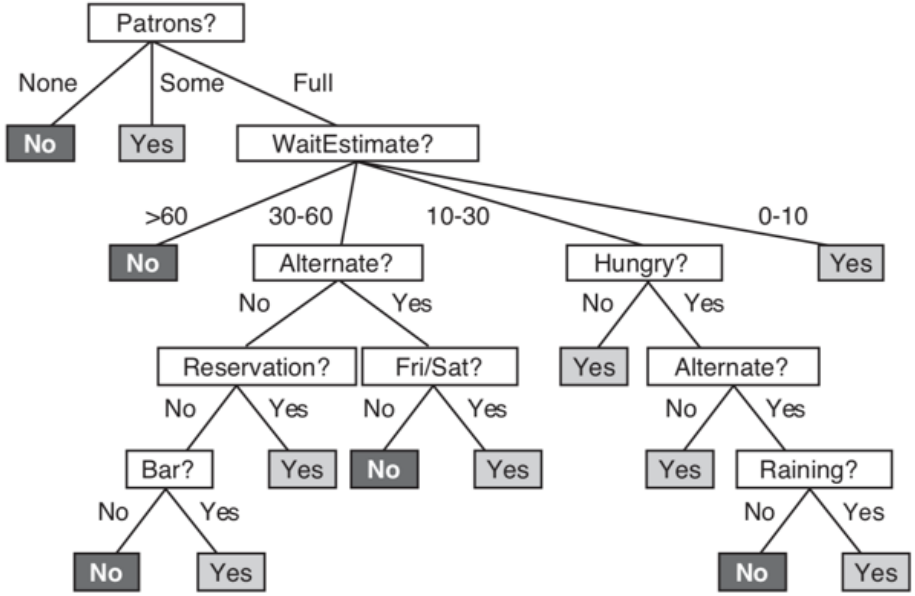
Many different consistent trees possible:

What quality is preferably?

More nodes v fewer nodes?

What are the consequences of having a deep tree with many nodes?





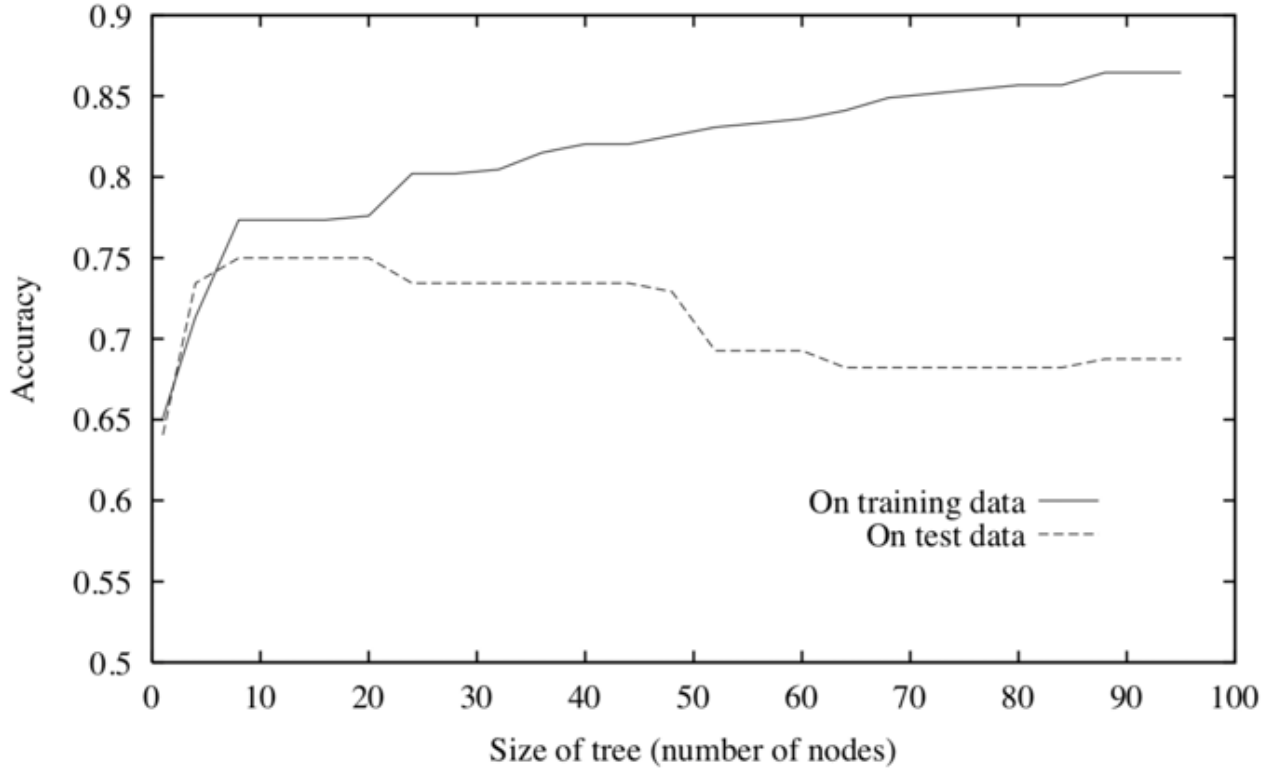
Inductive Bias of ID3 Algorithm

Shorter trees are preferred in ID3, trees with **high-information features closer to the root** are preferred

Biases allow us to learn, but you should understand what your algorithm's bias is.

Overfitting

- **Big idea**: You overfit if you do well on the training set, but not so well on the testing set.



Avoid Overfitting

- Make the tree smaller

- Some ideas on avoiding overly complex trees:
  - Stop growing when data split is not statistically significant
  - Grow full tree, then post-prune

Avoid Overfitting

- What are the benefits of decision trees compared to kNN?
- Disadvantages?
- When would you use one over the other?
  - if one column highly predicts the target variable → decision tree
  - If lots of predictors have similar weight in decision → kNN
  - If you must be able to interpret the data clearly → decision tree

Decision Tree Implementation with Scikit Learn Library

- `scikit-learn` is one of Python's main Machine Learning Libraries.

  "It is an open source machine learning library that supports supervised and unsupervised learning. It also provides various tools for model fitting, data preprocessing, model selection, model evaluation, and many other utilities."

  - built on NumPy, SciPy, and matplotlib
  - plays nicely with `pandas`
  - https://scikit-learn.org/stable/

Flow of Scikit Learn 'Algorithm'

- When working in Scikit Learn (`sklearn`), there is a general pattern that we can follow to implement any supported machine learning algorithm. It goes like this:

  - Load your data using `pd.read_csv()`
  - Split your data `train_test_split()`
  - Create your classifier/regressor object
  - Call `fit()` to train your model
  - Call `predict()` to get predictions
  - Call a metric function to measure the performance of your model

Let's jump into the Colab notebook

- Warm-Up Exercise
- DecisionTree from sklearn
  - classifier
  - Visualization

Extra sklearn activities (optional but encouraged)

- More exercises using sklearn library:
  - StandardScalar() for normalization
  - kNN, weighted kNN, decision tree from sklearn
    - Classifier vs. Regressor
- These exercises will be useful for your next two assignments (Notebook#4 as well as Project#1)