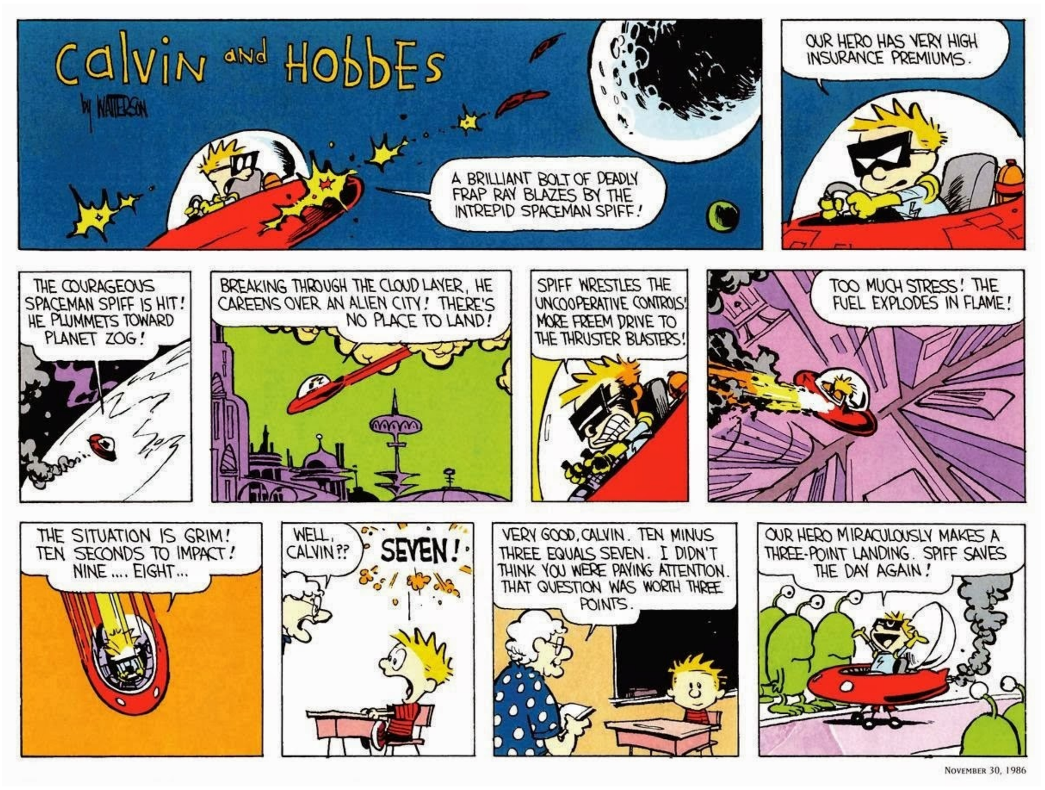
CSCI 3202

Lecture 33

November 12, 2025


Calvin and Hobbes
by WATTERSON
A BRILLIANT BOLT OF DEADLY FRAP RAY BLAZES BY THE INTREPID SPACEMAN SPIFF!
OUR HERO HAS VERY HIGH INSURANCE PREMIUMS.
THE COURAGEOUS SPACEMAN SPIFF IS HIT! HE PLUMMETS TOWARD PLANET ZOG!
BREAKING THROUGH THE CLOUD LAYER, HE CAREENS OVER AN ALIEN CITY! THERE'S NO PLACE TO LAND!
SPIFF WRESTLES THE UNCOOPERATIVE CONTROLS! MORE FREEM DRIVE TO THE THRUSTER BLASTERS!
TOO MUCH STRESS! THE FUEL EXPLODES IN FLAME!
THE SITUATION IS GRIM! TEN SECONDS TO IMPACT! NINE .... EIGHT...
WELL, CALVIN??
SEVEN!
VERY GOOD, CALVIN. TEN MINUS THREE EQUALS SEVEN. I DIDN'T THINK YOU WERE PAYING ATTENTION. THAT QUESTION WAS WORTH THREE POINTS.
OUR HERO MIRACULOUSLY MAKES A THREE-POINT LANDING. SPIFF SAVES THE DAY AGAIN!
NOVEMBER 30, 1986

Calvin and Hobbes by Bill Watterson. https://www.gocomics.com/calvinandhobbes

Announcements

- HW #9 will be released on Friday
    - Due after break
- Quiz #11 on Friday
    - Machine learning
    - Decision Trees
- Project due next week on Monday, November 17
    - There is a link to upload your project on Canvas
    - Both partners must submit a report
- Interview grading starting on Wednesday, November 19
    - We will have a sign up available on Monday

    - 15 minute interviews
    - Interviews are done on Zoom
    - Share your IDE
    - Run your project and answer questions
    - We will do interview grading after fall break as well
- You may resubmit Worked Out Problems (Q13-Q20) only from the midterm for regrading. The points you receive will replace your original score on the midterm. If you do not submit anything, there will be no change in your score. You have until Friday November 21 to submit. No late submissions will be accepted
    - Before calculating your final grade, I will add 5 points to your midterm exam score due to the length of the exam
    - These points will only count towards your midterm score. They will not count as extra credit toward your course grade

Machine Learning

- Machine learning can be broadly classified into **three categories**:
    - **Supervised Learning**
    - **Unsupervised Learning**
    - **Reinforcement Learning**

1. Supervised Learning

Supervised learning is the most common type of machine learning and is used when we have labeled data. In supervised learning, the algorithm learns from a **training set** where both the input data (features) and the correct outputs (labels) are provided.

- **Example Problem**: Predicting whether an email is spam or not. The model is trained on a dataset of emails (input) and whether each email is labeled as "spam" or "not spam" (output).

2. Unsupervised Learning

Unsupervised learning, in contrast, is used when the data does **not have labels**. The goal here is to identify hidden patterns or structures in the data without specific output values to guide the learning process.

- **Example Problem**: Segmenting customers into different groups based on purchasing behavior. The model isn’t told in advance what the groups should be, but it tries to discover distinct clusters of customer profiles.

Key Differences Between Supervised and Unsupervised Learning:

Feature	**Supervised Learning**	**Unsupervised Learning**
Data	Requires labeled data (input + output)	Uses unlabeled data (input only)
Goal	Predict or classify data based on labels	Discover hidden patterns or structures in the data
Example Algorithms	Decision Trees, Logistic Regression, SVM	K-Means, PCA, Hierarchical Clustering
Output	Prediction of output labels	Groupings, clusters, or associations

Decision Trees

A **decision tree** is a flowchart-like structure used to make decisions or predictions. Each internal node represents a "test" or "decision" based on an attribute (feature), and each branch represents an outcome of that test. The leaves of the tree contain the predictions (class labels for classification or values for regression).

- Decision Trees.pdf

How Decision Trees Work:

- Starting from the root node, the algorithm splits the data based on the feature that provides the best separation (e.g., minimizing variance or maximizing information gain).
- This process continues recursively, creating subtrees, until the data is perfectly separated, or some stopping criterion is reached (like a maximum depth or minimum number of samples per leaf).

Example: Suppose we want to predict whether a person buys a product based on their age and income level. The decision tree might first split the data by age, then further split by income, leading to a prediction about whether the person will buy the product.

Pros of Decision Trees:

- Easy to understand and visualize.
- Can handle both numerical and categorical data.
- Non-linear, so can capture complex patterns.

Cons of Decision Trees:

- Prone to overfitting if the tree is too deep.
- Sensitive to small variations in the data.

How to fit a Decision Tree

A decision tree is a supervised learning algorithm used for classification and regression tasks. It recursively splits data into subsets based on feature values, resulting in a tree-like structure with decision nodes and leaf nodes. At each decision node, a feature is tested, and based on the test result, the data is split into two or more branches. The tree continues to grow until a stopping criterion is met (e.g., maximum depth, minimum samples per leaf, etc.).

Key Components of a Decision Tree:

- Root Node: The top node of the tree, which represents the entire dataset.
- Internal Nodes: Nodes where the data is split based on feature values.
- Leaf Nodes: Terminal nodes that represent the predicted output (class label for classification, value for regression).
- Branches: The paths that connect nodes, representing the decisions based on feature tests.

Fitting a Decision Tree

The process of fitting a decision tree involves splitting the data recursively based on the best feature at each node. The goal is to select the feature that best separates the data, using a criterion like Gini Impurity or Entropy for classification, or Mean Squared Error (MSE) for regression.

Example of Fitting a Decision Tree

We use a simple dataset for binary classification:

Feature 1	**Feature 2**	**Class Label**

2	1	0
3	2	0
5	3	1
6	4	1
7	5	1

Use Gini Impurity as the splitting criterion. The Gini Impurity for a set of classes is calculated as:

$$Gini(S) = 1 - \sum_{i=1}^{C} p_i^2$$

Where:

- p_i is the proportion of samples in class i
- C is the number of classes

Step 1: Choose the Best Split

For each feature, we calculate the Gini Impurity for every possible split, choosing the one with the lowest Gini value.

- Feature 1: If we split at 5, we get:
- Left (Feature 1 ≤ 5): Class 0 (samples 1, 2)
  - Right (Feature 1 > 5): Class 1 (samples 3, 4, 5)
  - Calculate Gini for both groups and find the best split.

Step 2: Create the Decision Tree

- Based on the best split, we partition the dataset into subsets. For each subset, we repeat the process until all data points are classified or the stopping criterion is met.

Step 3: Stop Criteria

Stopping criteria could include:

- Maximum tree depth
- Minimum number of samples required to split a node
- Minimum impurity decrease

Model Diagnostics
Once a decision tree is built, it’s important to assess how well it performs. This involves evaluating its accuracy on unseen data using cross-validation and measuring its performance metrics

Common Diagnostics:

- Accuracy: The percentage of correctly classified instances.
- Confusion Matrix: A table that describes the performance of a classification algorithm.
- Precision, Recall, and F1-Score: Useful in imbalanced datasets.
- ROC Curve & AUC: For evaluating classifier performance across different thresholds.

Measures of Fit
- Adapted from ChatGPT
- Measures of fit describe various aspects of how well a model fits our data, either the training or test data

---

- In **machine learning**, **Gini impurity** (or just "Gini") is a metric used to measure how *pure* or *impure* a dataset is — especially in **decision trees**

The Gini impurity for a node is defined as:

$$Gini = 1 - \sum_{i=1}^{C} p_i^2$$

Where:

- C is the number of classes,
- p_i is the probability (proportion) of a data point belonging to class (i) in that node.

Interpretation:

- If all elements belong to **one class** → Gini = **0** (pure).
- If classes are perfectly mixed (e.g., 50/50 in binary classification) → Gini = **0.5**.
- The **higher the Gini**, the **more impure** the node is.

Example:
Suppose a node contains:

- 4 samples of class A
- 6 samples of class B

Then:

$$p_A = \frac{4}{10} = 0.4, \quad p_B = \frac{6}{10} = 0.6$$

$$Gini = 1 - (0.4)^2 - (0.6)^2 = 1 - 0.16 - 0.36 = 0.48$$

Use in Decision Trees:
During training, the decision tree algorithm:

- Tries different splits
- Calculates the **Gini impurity** for each

- Chooses the split that results in the **lowest Gini** (i.e., more pure child nodes)

In **machine learning**, especially in **classification tasks**, **precision** is a metric that tells you how *accurate your positive predictions* are.

Definition of Precision:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

Where:

- **True Positives (TP)**: Correctly predicted positives
- **False Positives (FP)**: Incorrectly predicted positives (model said “yes” but the answer was “no”)

What It Tells You:
Precision answers the question:

> *"**Of all the times the model said 'yes', how many times was it actually right?**"*

Example:
Imagine you're building a loan approval model:

- The model approved 100 applications.
- Out of those, only 80 were actually good (true positives).

- 20 were bad approvals (false positives).

Then:

$$\text{Precision} = \frac{80}{80+20} = \frac{80}{100} = 0.8$$

So the model has **80% precision —** when it predicts a loan should be approved, it's right 80% of the time.

When to Focus on Precision?
Use **precision** when **false positives are costly**.

- Email spam filters: Better to not label a good email as spam (high precision).
- Medical diagnosis: Don't falsely tell someone they're sick.

Definition of Accuracy:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} = \frac{TP+TN}{TP+TN+FP+FN}$$

Where:

- **TP** = True Positives
- **TN** = True Negatives
- **FP** = False Positives
- **FN** = False Negatives

In Plain Terms:

> *"Out of* ***all*** *the predictions the model made, how many did it get right?"*

Example:
Say you have a loan approval classifier, and it evaluated 100 applicants:

- 70 were correctly classified (50 approved + 20 denied)
- 30 were misclassified (15 wrongly approved + 15 wrongly denied)

Then:

$$\text{Accuracy} = \frac{70}{100} = 0.70 \text{ or } 70\%$$

When Accuracy Can Be Misleading:
If the data is **imbalanced**, accuracy can give a false sense of performance.

Example:

- Dataset: 95% of applicants are denied loans.
- Model predicts "denied" for everyone → **95% accuracy**, but it's actually **useless**.

That's why we often look at **precision**, **recall**, **F1-score**, or use **confusion matrices** for more insight.

Decision Tree Example

- Decision Tree Example.ipynb

Upcoming

- Model Fitting
- Neural Networks