Decision Trees

Zach Gulde

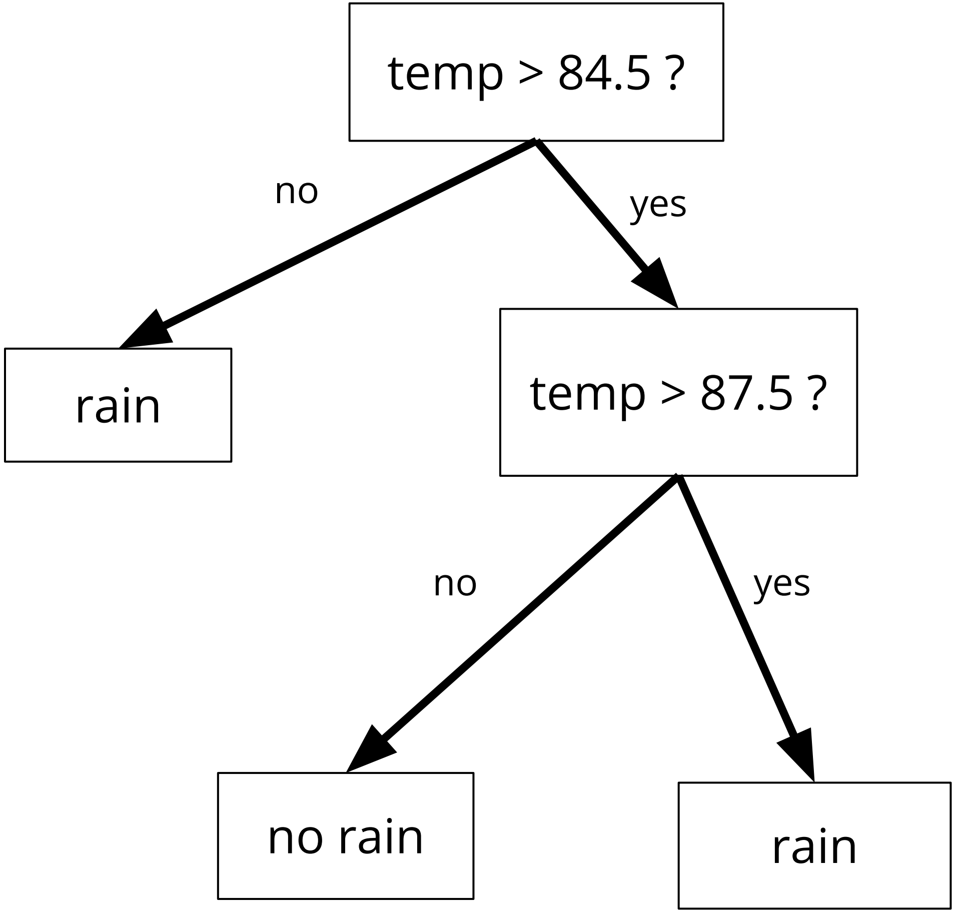
Topics

1. What are decision trees?
2. How do decision trees work?
3. How do we implement decision trees?

Decision Trees

- A set of rules for making predictions
- A series of yes/no questions are asked at each **node** until the end, a **leaf**, is hit

- Pro: Easy to interpret
- Pro: fast to make predictions
- Pro: minimal preprocessing needed
- Con: doesn't consider feature interactions
- Con: complex to train
- Con: prone to overfitting



How do we make Decision Trees?

Warning

Theoretical concepts and some math notation follows.

In practice this is all implemented by scikit-learn.

Tree-Growing Algorithm

1. Calculate impurity for current node, *and* each feature
2. If the current split has the lowest impurity, then this is a leaf node (end point)
3. Else choose the feature with the lowest impurity and split into 2 nodes
4. Repeat for each remaining node

To calculate impurity for a feature:

- for a binary categorical feature, split and calculate impurity
- otherwise calculate all possible splits; the one with lowest impurity is the split for this feature

Gini Impurity Algorithm

For a leaf, gini impurity (*G*) is

for each class *i* in *k* total classes

$$G = 1 - \sum_{i}^{k} p_i^2$$

For a given split, weighted Gini (G_w) is

$$G_w = \frac{\sum_i^k n_i G_i}{\sum_i^k n_i}$$

For binary classification, this simplifies to

$$G = 1 - p^2 - q^2$$

$$G_w = \frac{n_1 G_1 + n_2 G_2}{n_1 + n_2}$$

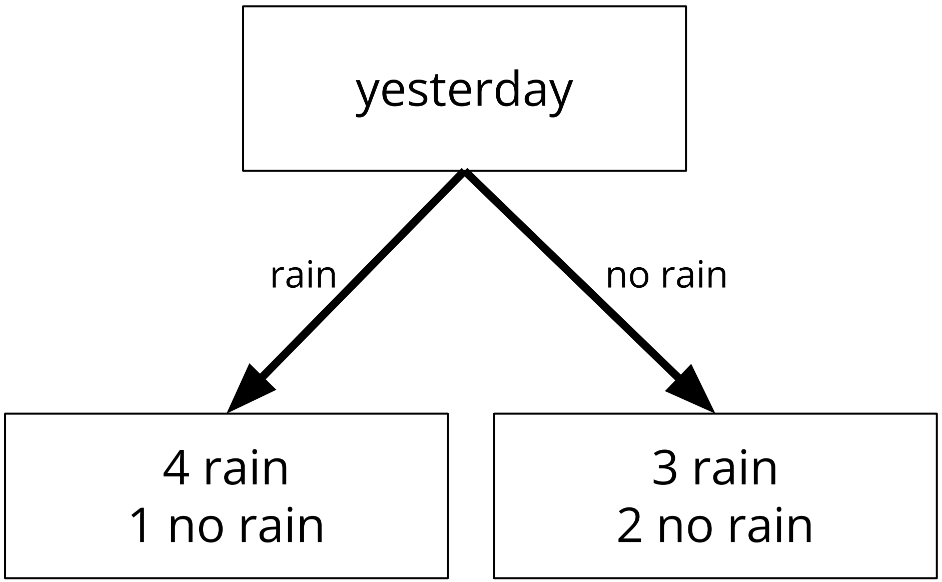
We want to predict whether or not it's going to rain today.

	temp	yesterday	today
0	89	no rain	rain
1	86	rain	no rain
2	81	rain	rain
3	80	no rain	rain
4	81	rain	rain
5	89	rain	rain
6	80	no rain	rain
7	80	no rain	no rain
8	89	no rain	no rain
9	83	rain	rain

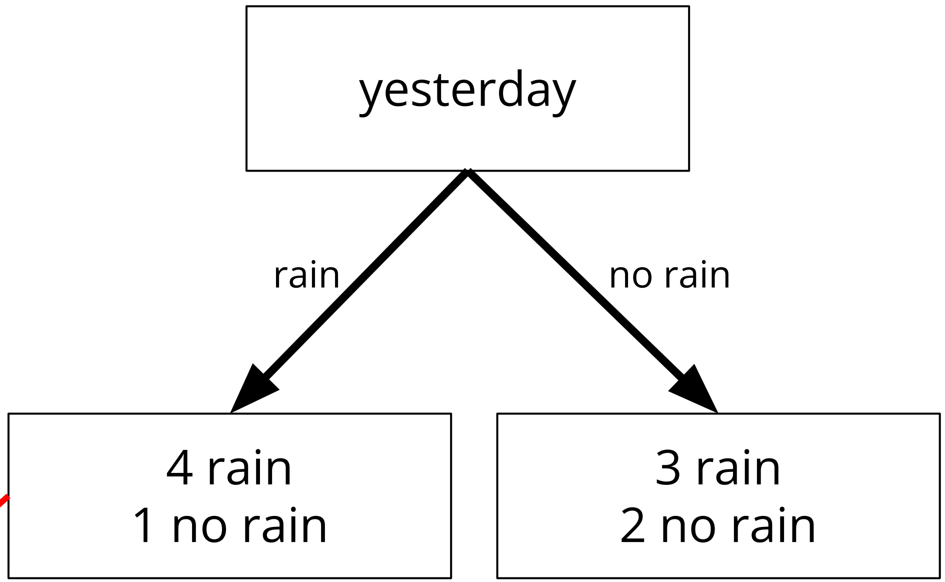
	temp	yesterday	today
0	89	no rain	rain
3	80	no rain	rain
6	80	no rain	rain
7	80	no rain	no rain
8	89	no rain	no rain
1	86	rain	no rain
2	81	rain	rain
4	81	rain	rain
5	89	rain	rain
9	83	rain	rain

Start by examining yesterday's weather to predict today's. What's the gini impurity?

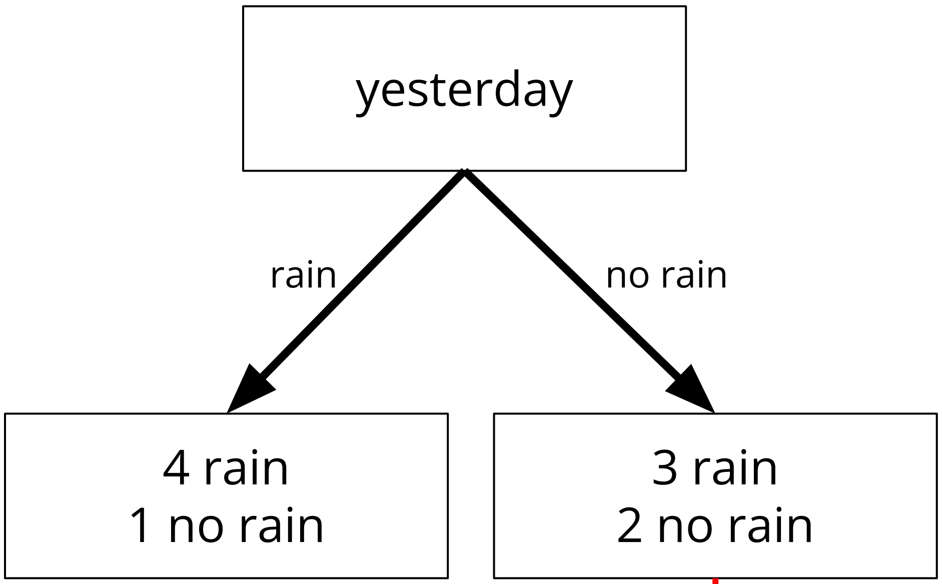
	temp	yesterday	today
0	89	no rain	rain
3	80	no rain	rain
6	80	no rain	rain
7	80	no rain	no rain
8	89	no rain	no rain
1	86	rain	no rain
2	81	rain	rain
4	81	rain	rain
5	89	rain	rain
9	83	rain	rain



	temp	yesterday	today
0	89	no rain	rain
3	80	no rain	rain
6	80	no rain	rain
7	80	no rain	no rain
8	89	no rain	no rain
1	86	rain	no rain
2	81	rain	rain
4	81	rain	rain
5	89	rain	rain
9	83	rain	rain



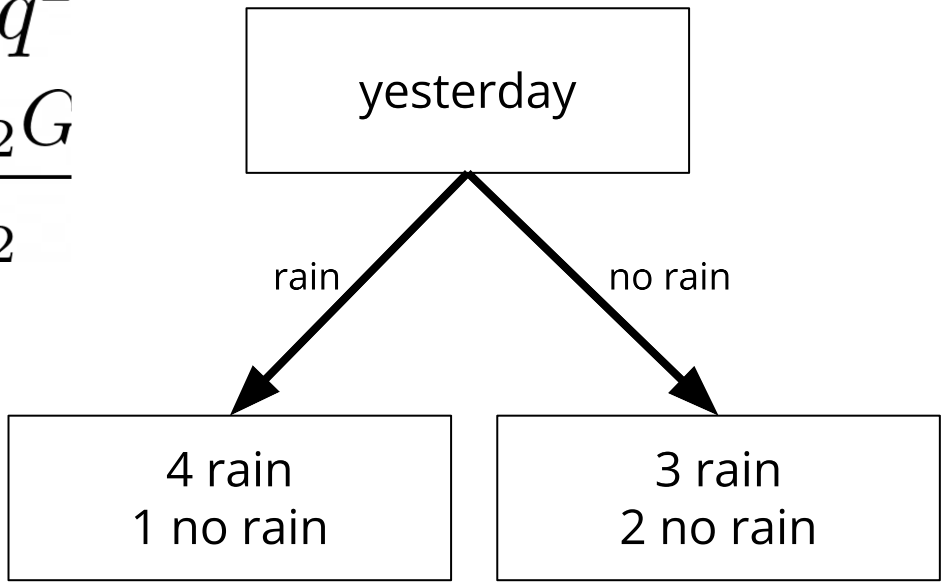
	temp	yesterday	today
0	89	no rain	rain
3	80	no rain	rain
6	80	no rain	rain
7	80	no rain	no rain
8	89	no rain	no rain
1	86	rain	no rain
2	81	rain	rain
4	81	rain	rain
5	89	rain	rain
9	83	rain	rain



$$G = 1 - p^2 - q^2$$

$$G_w = \frac{n_1 G_1 + n_2 G}{n_1 + n_2}$$

	temp	yesterday	today
0	89	no rain	rain
3	80	no rain	rain
6	80	no rain	rain
7	80	no rain	no rain
8	89	no rain	no rain
1	86	rain	no rain
2	81	rain	rain
4	81	rain	rain
5	89	rain	rain
9	83	rain	rain



$G_1 = 1 - (4/5)^2 - (1/5)^2 = .320$

$G_2 = 1 - (3/5)^2 - (2/5)^2 = .480$

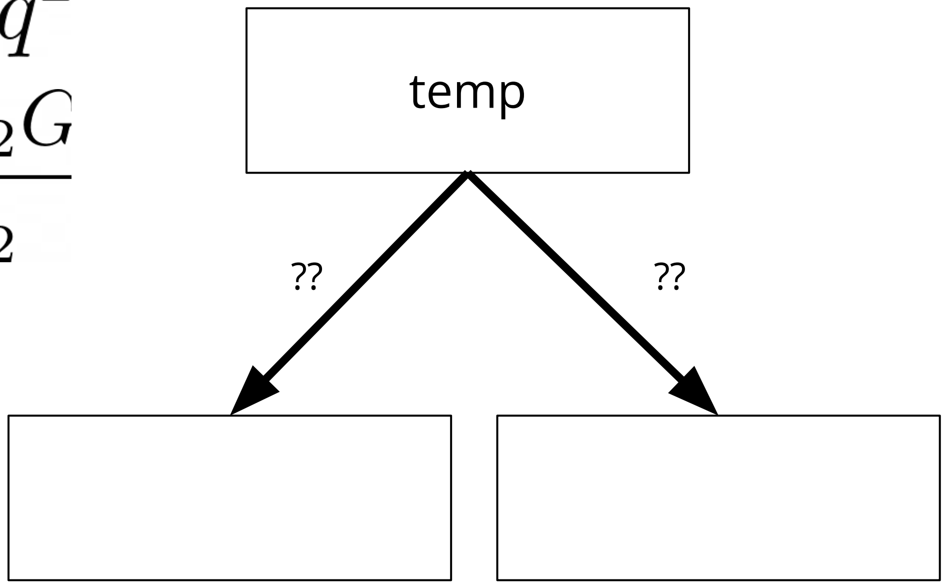
$G_w = (5(.320) + 5(.480)) / (5 + 5) = .400$

$$G = 1 - p^2 - q^2$$

$$G_w = \frac{n_1 G_1 + n_2 G}{n_1 + n_2}$$

.400

	yesterday	temp	today
3	no rain	80	rain
6	no rain	80	rain
7	no rain	80	no rain
2	rain	81	rain
4	rain	81	rain
9	rain	83	rain
1	rain	86	no rain
0	no rain	89	rain
5	rain	89	rain
8	no rain	89	no rain



$$G = 1 - p^2 - q^2$$

$$G_w = \frac{n_1 G_1 + n_2 G}{n_1 + n_2}$$

.400

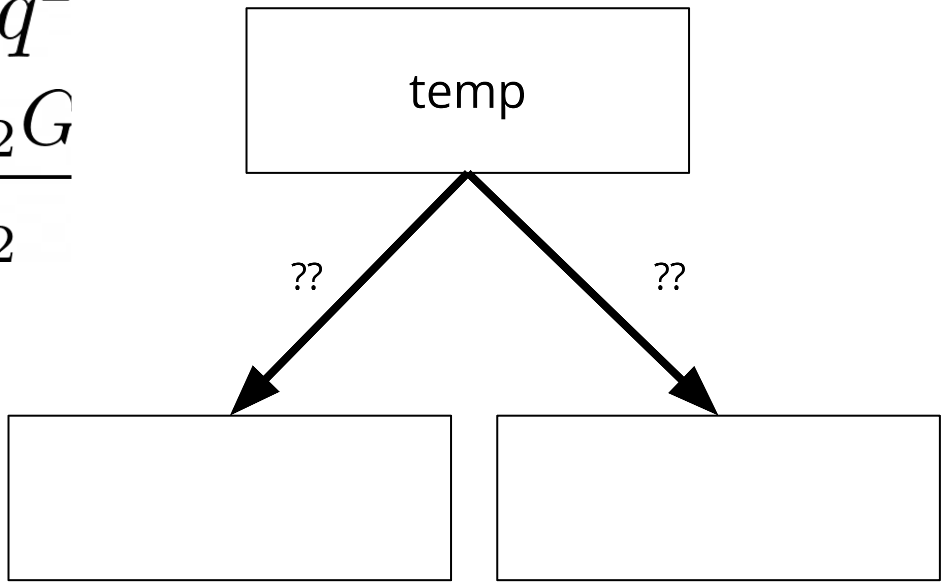
	yesterday	temp	today
3	no rain	80	rain
6	no rain	80	rain
7	no rain	80	no rain
2	rain	81	rain
4	rain	81	rain
9	rain	83	rain
1	rain	86	no rain
0	no rain	89	rain
5	rain	89	rain
8	no rain	89	no rain

80.5

82

84.5

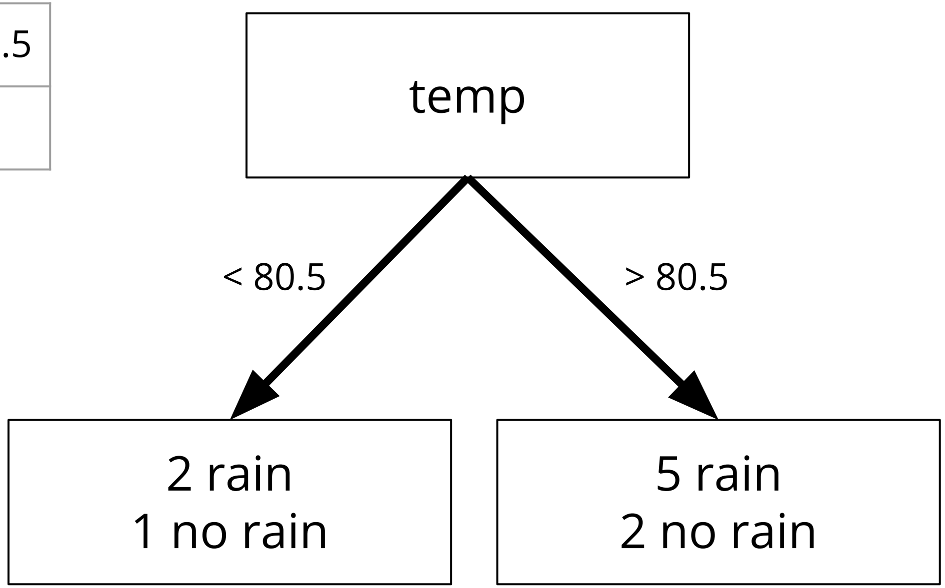
87.5



cutoff	80.5	82	84.5	87.5
G	.419			

.400

	yesterday	temp	today
3	no rain	80	rain
6	no rain	80	rain
7	no rain	80	no rain
2	rain	81	rain
4	rain	81	rain
9	rain	83	rain
1	rain	86	no rain
0	no rain	89	rain
5	rain	89	rain
8	no rain	89	no rain



$$G_1 = 1 - ({}^{2}/_{3})^2 - ({}^{1}/_{3})^2 = .444$$

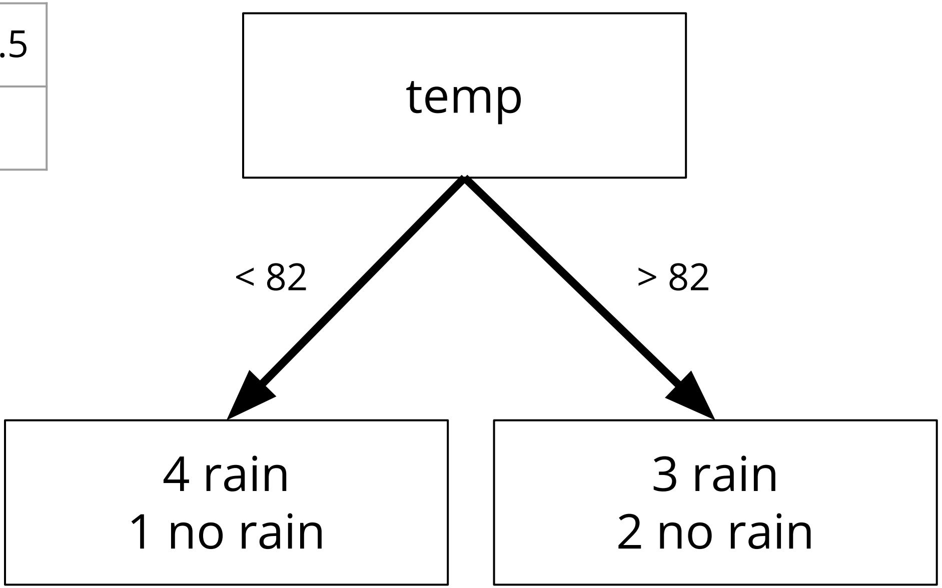
$$G_2 = 1 - ({}^{5}/_{7})^2 - ({}^{2}/_{7})^2 = .408$$

$$G_w = (\ 3(.444) + 7(.408)\ ) \ /\ (3 + 7) = .419$$

cutoff	80.5	82	84.5	87.5
G	.419	.400		

.400

	yesterday	temp	today
3	no rain	80	rain
6	no rain	80	rain
7	no rain	80	no rain
2	rain	81	rain
4	rain	81	rain
9	rain	83	rain
1	rain	86	no rain
0	no rain	89	rain
5	rain	89	rain
8	no rain	89	no rain



$$G_1 = 1 - (^4/_5)^2 - (^1/_5)^2 = .320$$

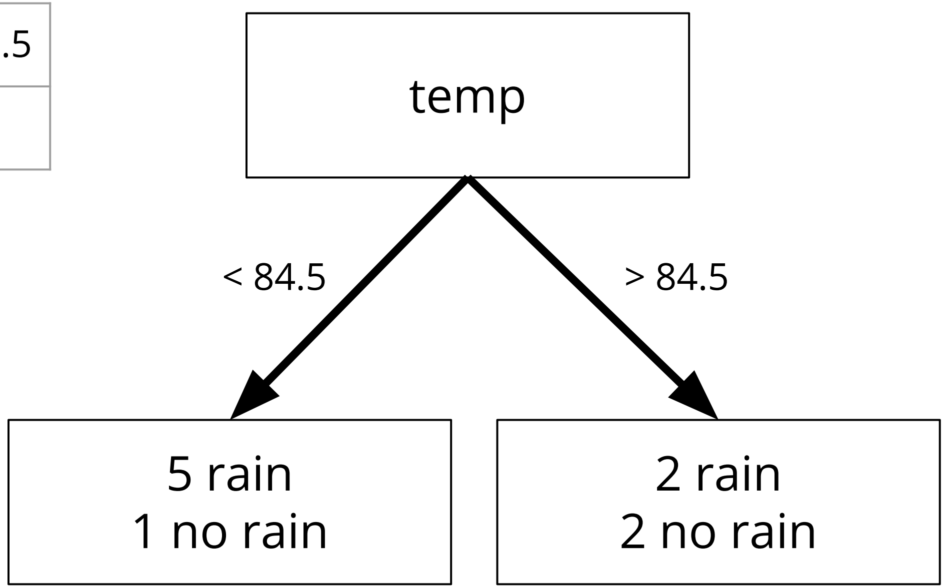
$$G_2 = 1 - (^3/_5)^2 - (^2/_5)^2 = .480$$

$$G_w = (\ 5(.320) + 5(.480)\ ) \ / \ (5 + 5) = .400$$

cutoff	80.5	82	84.5	87.5
G	.419	.400	.366	

.400

	yesterday	temp	today
3	no rain	80	rain
6	no rain	80	rain
7	no rain	80	no rain
2	rain	81	rain
4	rain	81	rain
9	rain	83	rain
1	rain	86	no rain
0	no rain	89	rain
5	rain	89	rain
8	no rain	89	no rain



$$G_1 = 1 - (^5/_6)^2 - (^1/_6)^2 = .277$$

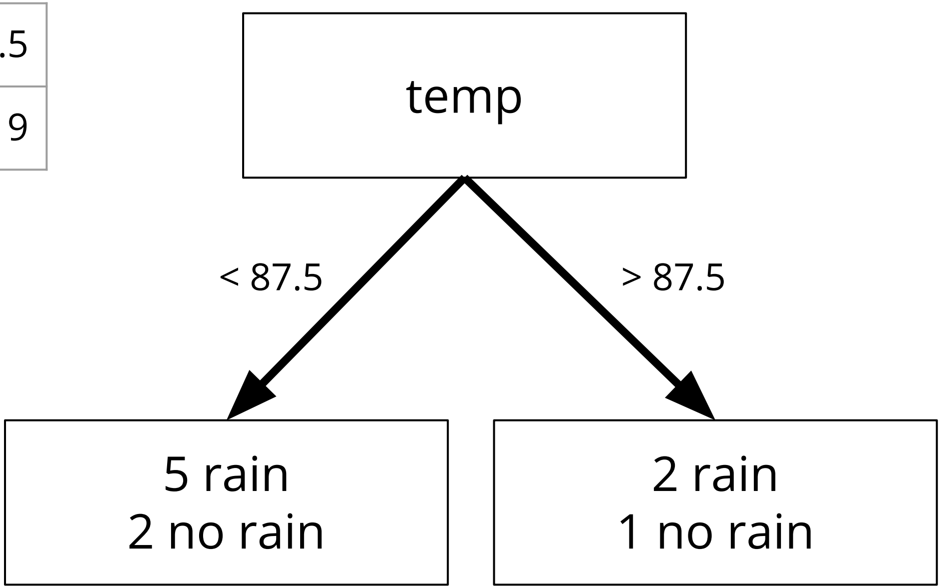
$$G_2 = 1 - (^2/_4)^2 - (^2/_4)^2 = .500$$

$$G_w = (\ 6(.277) + 4(.500)\ ) \ / \ (6 + 4) = .366$$

cutoff	80.5	82	84.5	87.5
G	.419	.400	.366	.419

.400

	yesterday	temp	today
3	no rain	80	rain
6	no rain	80	rain
7	no rain	80	no rain
2	rain	81	rain
4	rain	81	rain
9	rain	83	rain
1	rain	86	no rain
0	no rain	89	rain
5	rain	89	rain
8	no rain	89	no rain



$$G_1 = 1 - (^5/_7)^2 - (^2/_7)^2 = .408$$

$$G_2 = 1 - (^2/_3)^2 - (^1/_3)^2 = .444$$

$$G_w = (\ 7(.408) + 3(.444)\ ) / (7 + 3) = .419$$

cutoff	80.5	82	84.5	87.5
G	.419	.400	.366	.419

temp

.420

.400

	yesterday	temp	today
3	no rain	80	rain
6	no rain	80	rain
7	no rain	80	no rain
2	rain	81	rain
4	rain	81	rain
9	rain	83	rain
1	rain	86	no rain
0	no rain	89	rain
5	rain	89	rain
8	no rain	89	no rain

Without any splitting, we currently have a Gini Impurity of .420

$$G = 1 - ({}^{3}/_{10})^2 - ({}^{7}/_{10})^2 = .420$$

cutoff	80.5	82	84.5	87.5
G	.419	.400	.366	.419

temp

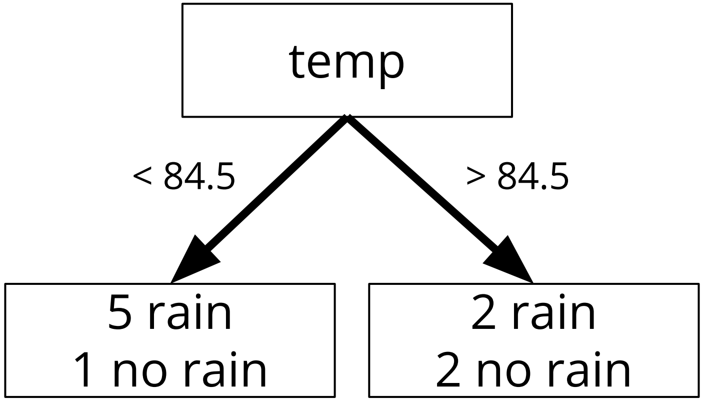
.420

.400

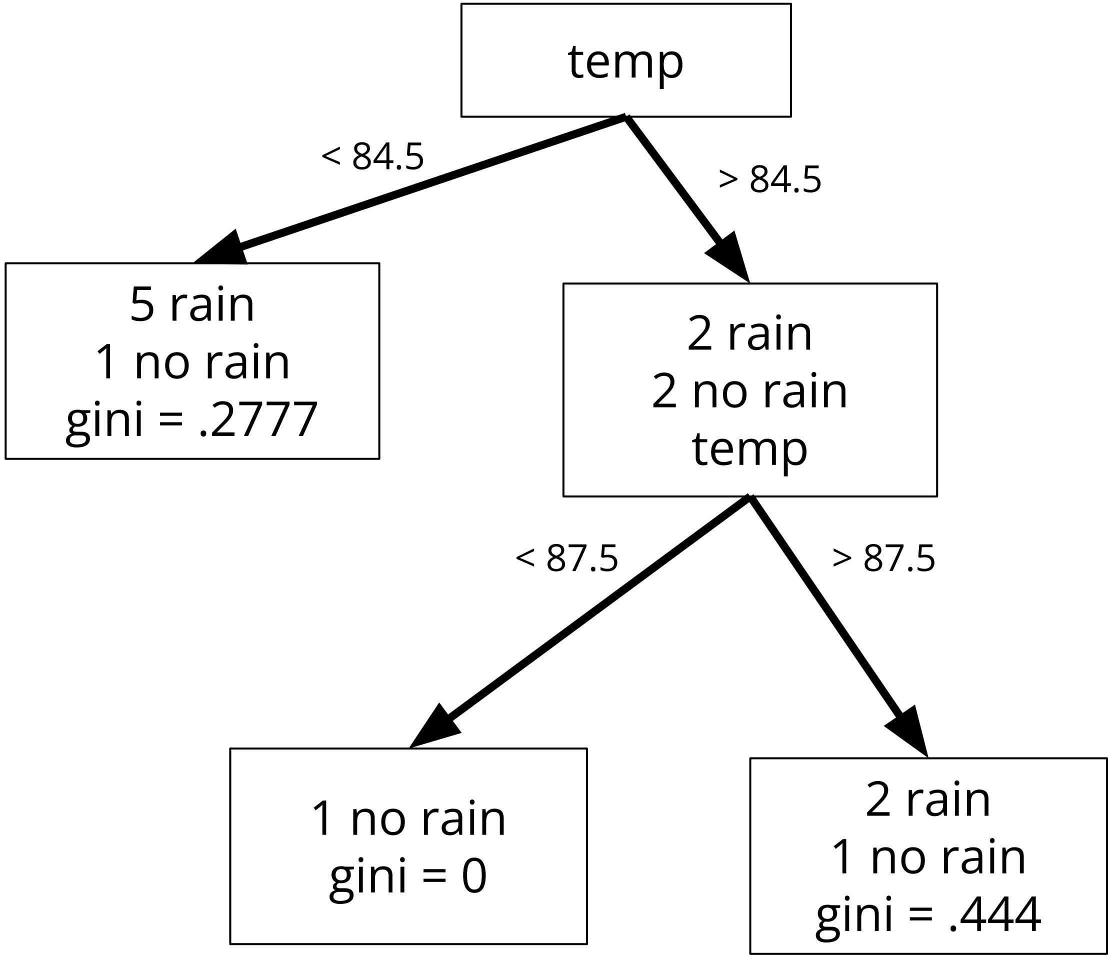
	yesterday	temp	today
3	no rain	80	rain
6	no rain	80	rain
7	no rain	80	no rain
2	rain	81	rain
4	rain	81	rain
9	rain	83	rain
1	rain	86	no rain
0	no rain	89	rain
5	rain	89	rain
8	no rain	89	no rain

Splitting based on temp with a cutoff point of 84.5 gives us the lowest impurity, so we will split at this point.

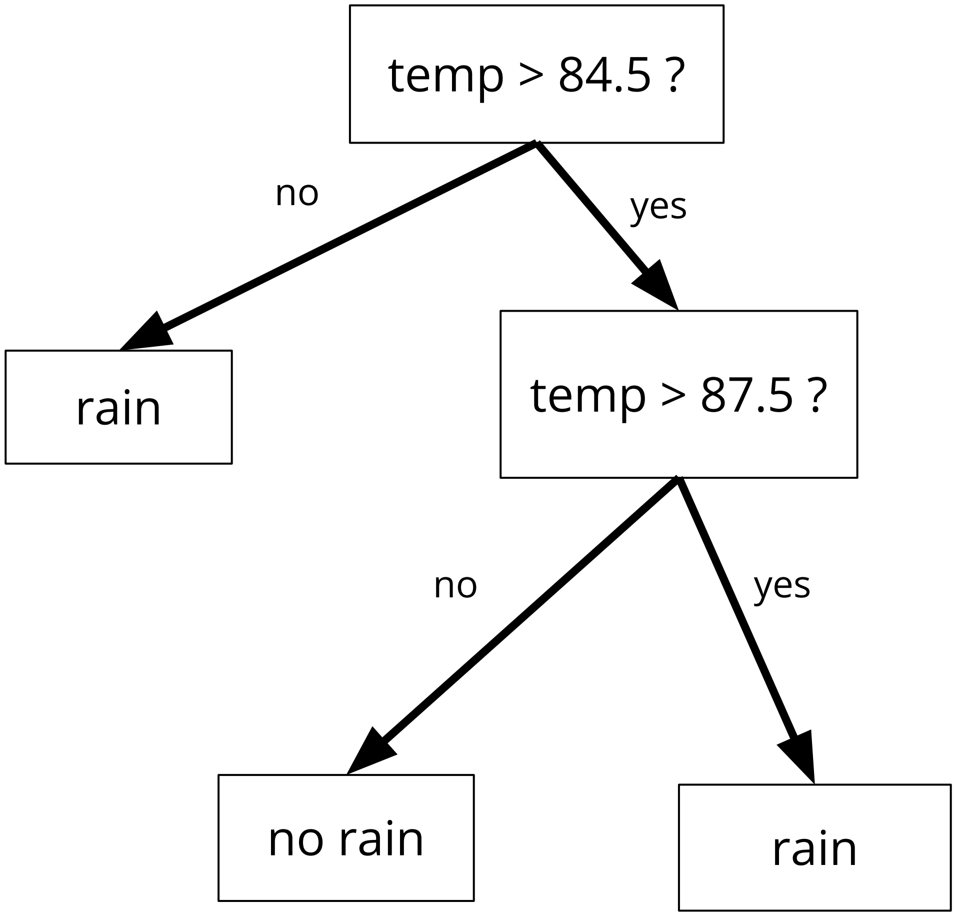
	yesterday	temp	today
3	no rain	80	rain
6	no rain	80	rain
7	no rain	80	no rain
2	rain	81	rain
4	rain	81	rain
9	rain	83	rain
1	rain	86	no rain
0	no rain	89	rain
5	rain	89	rain
8	no rain	89	no rain



	yesterday	temp	today
3	no rain	80	rain
6	no rain	80	rain
7	no rain	80	no rain
2	rain	81	rain
4	rain	81	rain
9	rain	83	rain
1	rain	86	no rain
0	no rain	89	rain
5	rain	89	rain
8	no rain	89	no rain



	yesterday	temp	today	predicted
0	no rain	89	rain	rain
1	rain	86	no rain	no rain
2	rain	81	rain	rain
3	no rain	80	rain	rain
4	rain	81	rain	rain
5	rain	89	rain	rain
6	no rain	80	rain	rain
7	no rain	80	no rain	rain
8	no rain	89	no rain	rain
9	rain	83	rain	rain



Hyperparams

Decision Tree Hyperparameters

- `max_depth`: maximum number of splits (default: None)
- `min_samples_split`: minimum number of data points required to split a node (default: 2)
- `min_samples_leaf`: minimum number of data points required to be present in a endpoint or leaf (default: 1)
- `max_leaf_nodes`: maximum number of "endpoints", or leaves (default: None)

Implementation

Process Overview

1. Imports
2. Split Data
3. Create models
    a. Create
    b. fit
    c. .predict / .score
4. Evaluate on validate
5. Interpret
6. Evaluate on test

Process Overview

1. Imports
2. Split Data
3. Create models
    a. Create
    b. fit
    c. .predict / .score
4. Evaluate on validate
5. Interpret
6. Evaluate on test

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import plot_tree, export_text
from sklearn.metrics import classification_report
```

Process Overview

1. Imports
2. Split Data
3. Create models
    a. Create
    b. fit
    c. .predict / .score
4. Evaluate on validate
5. Interpret
6. Evaluate on test

```
train, test, validate

X_train, y_train
X_validate, y_validate
X_test, y_test
```

Process Overview

1. Imports
2. Split Data
3. Create models
    a. Create the model
    b. fit it
    c. use it - .predict / .score
4. Evaluate on validate
5. Interpret
6. Evaluate on test

```
# a
model = DecisionTreeClassifier(max_depth=1)
# b
model.fit(X_train)
# c
model.score(X_train, y_train) # accuracy
# or
train['prediction'] = model.predict(X_train)
classification_report(
    train.actual,
    train.prediction,
)
```

Process Overview

1. Imports
2. Split Data
3. Create models
    a. Create the model
    b. fit it
    c. use it - .predict / .score
4. Evaluate on validate
5. Interpret
6. Evaluate on test

```
model1.score(X_validate, y_validate)
model2.score(X_validate, y_validate)
model3.score(X_validate, y_validate)
# or
predictions1 = model1.predict(X_validate)
classification_report(
    validate.actual, predictions1
)
...
```

Process Overview

1. Imports
2. Split Data
3. Create models
    a. Create the model
    b. fit it
    c. use it - .predict / .score
4. Evaluate on validate
5. Interpret
6. Evaluate on test

```
model.feature_importances_

print(export_text(model))

plot_tree(model)
```

Process Overview

1. Imports
2. Split Data
3. Create models
   a. Create the model
   b. fit it
   c. use it - .predict / .score
4. Evaluate on validate
5. Interpret
6. Evaluate on test

```
best_model.score(X_test)
# or
pred = best_model.predict(X_test)
classification_report(
    test.actual, pred
)
```