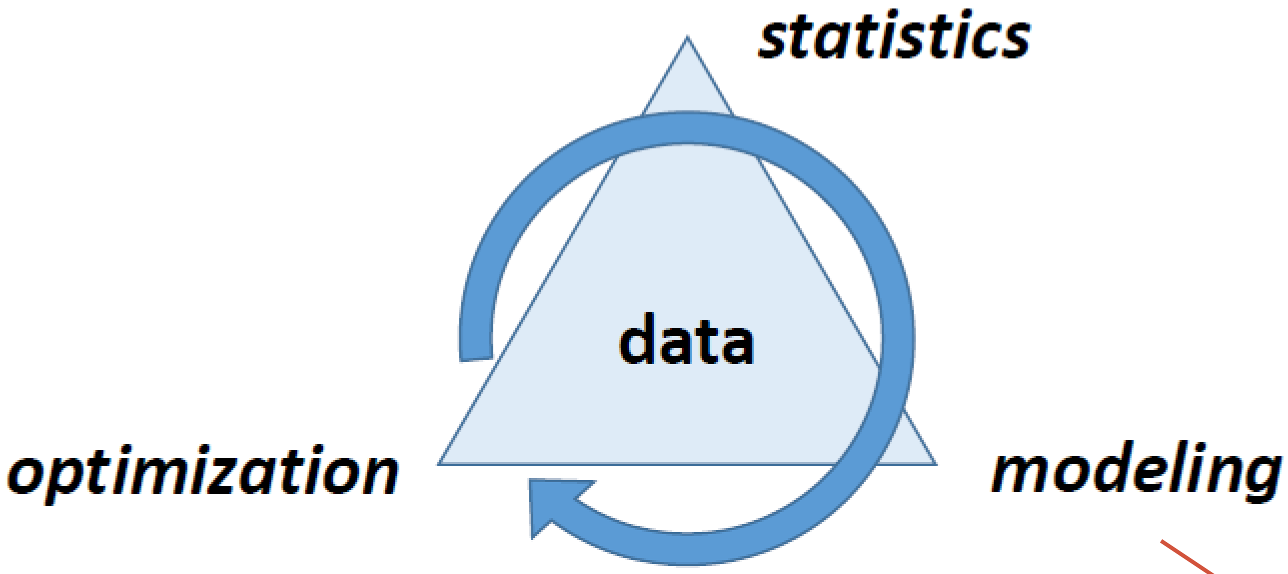
DECISION TREES

By Itay Evron

Outline

- Decision trees – model class
- How to build a tree?
  - Greedy algorithms
  - Dry example
  - Wet example
- Overfitting and model complexity

Three pillars of learning

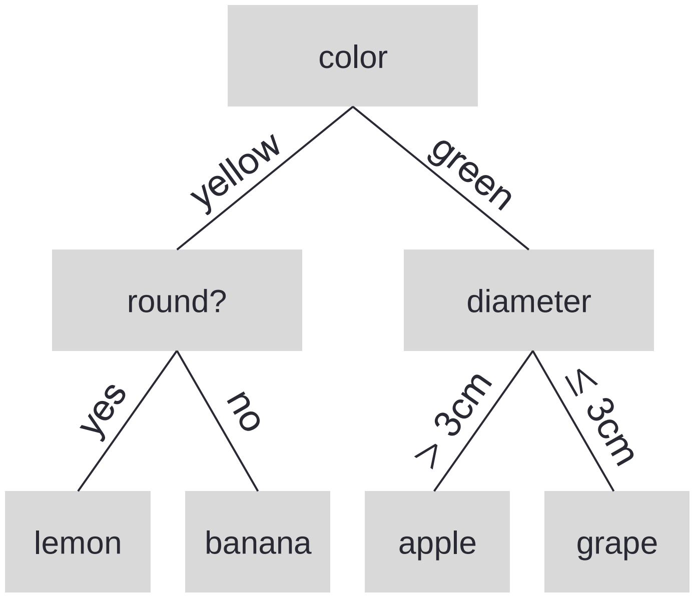


What is a decision tree?

Decision Trees

- Leaf
  - Class or a distribution over classes
- Branches / edges
  - “Questions” on features
- Depth
  - The length of the longest path (in edges)
  - We do not count the root
  - E.g., the depth on the right is 2

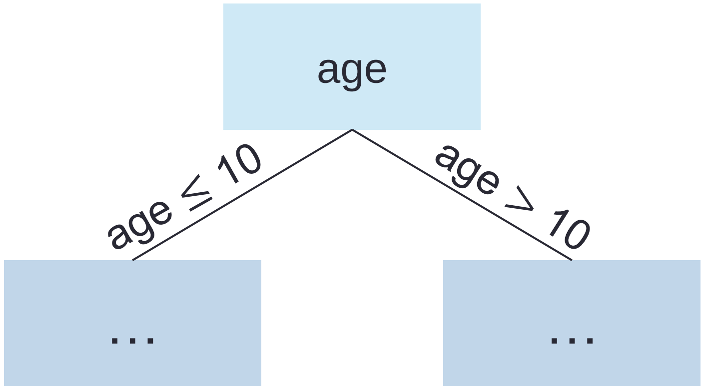
Example: Fruit classifier



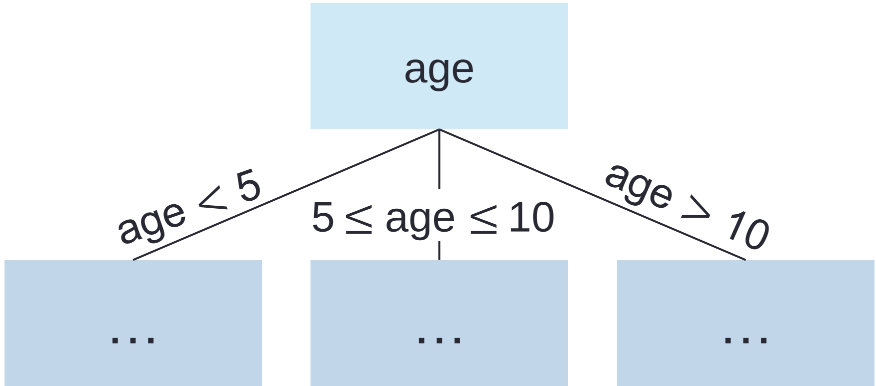
Clarification

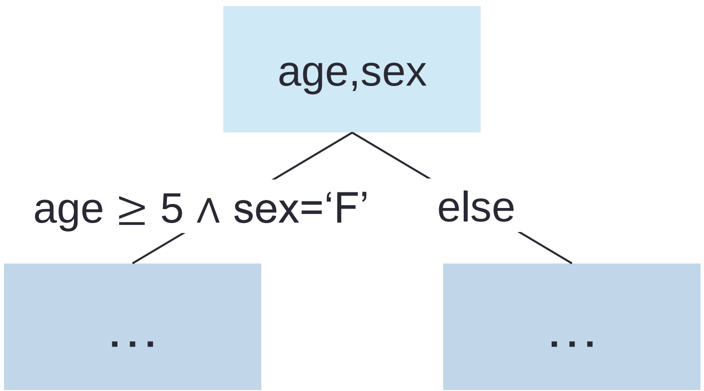
- In this course, we only:
  - deal with “binary” trees.
  - use <u>one</u> feature/attribute per node (thresholds / equalities).

Yes

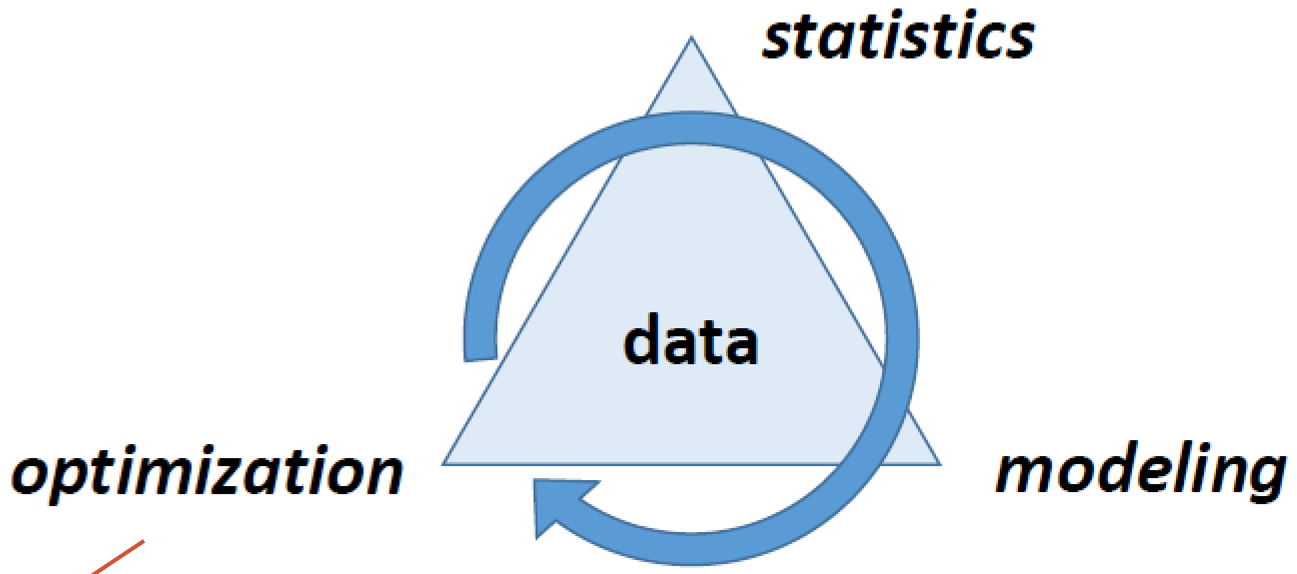


No





Three pillars of learning



How to build/find a “good” decision tree?

Optimization

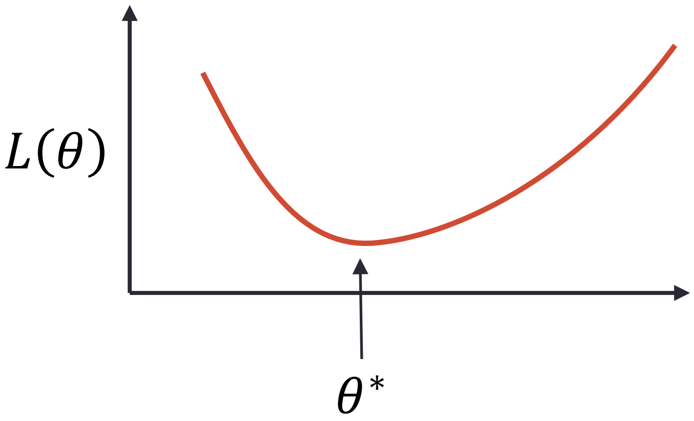
loss of hypothesis

$$\min_{\theta \in \Theta} L(\theta)$$

hypothesis class

- Goal: find a hypothesis with lowest loss / risk.

- Linear classifiers (next tutorial)
  - Continuous optimization: Optimize over real numbers.
  - Problem is often differentiable.



- Decision trees
  - Discrete optimization: Optimize over decision trees.
  - Not differentiable.

?

Greedy algorithms

- Optimizing over tree classes is hard.
  - Not differentiable
  - Exponentially many trees
  - Cannot perform ERM by inspecting all possible trees.

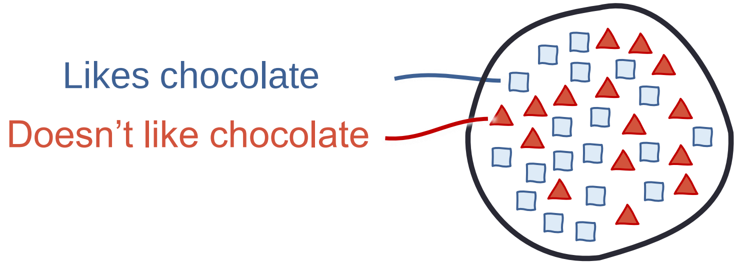
- Greedy algorithms are a popular alternative.

- Top-Down Induction of Decision Trees (TDIDT)
  - Start with all samples.
  - Iteratively choose splitting rules to partition samples.

- Splitting criteria
  - Leading principle: simplicity.
  - At node v, split so as to decrease impurity the most.

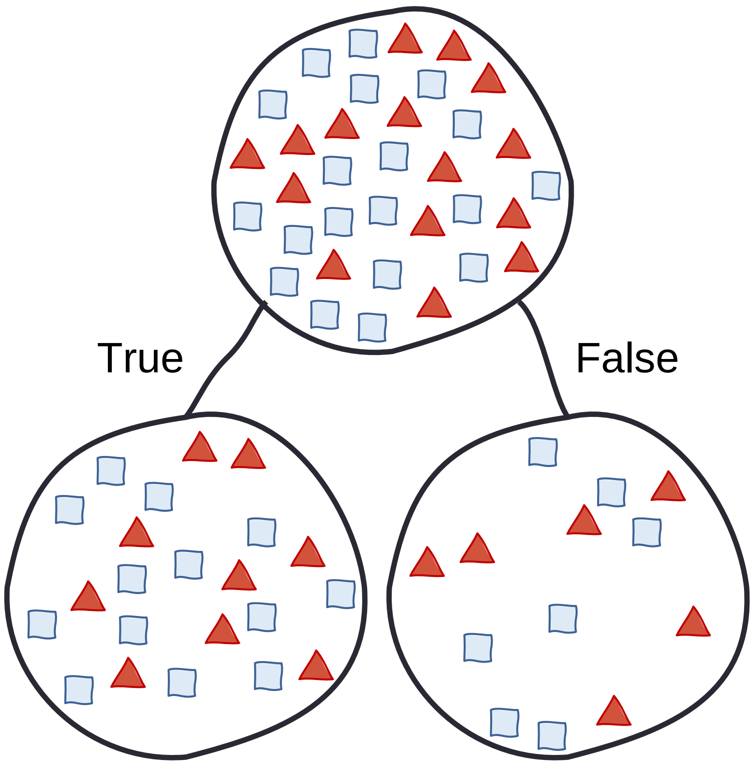
Split illustration

Target

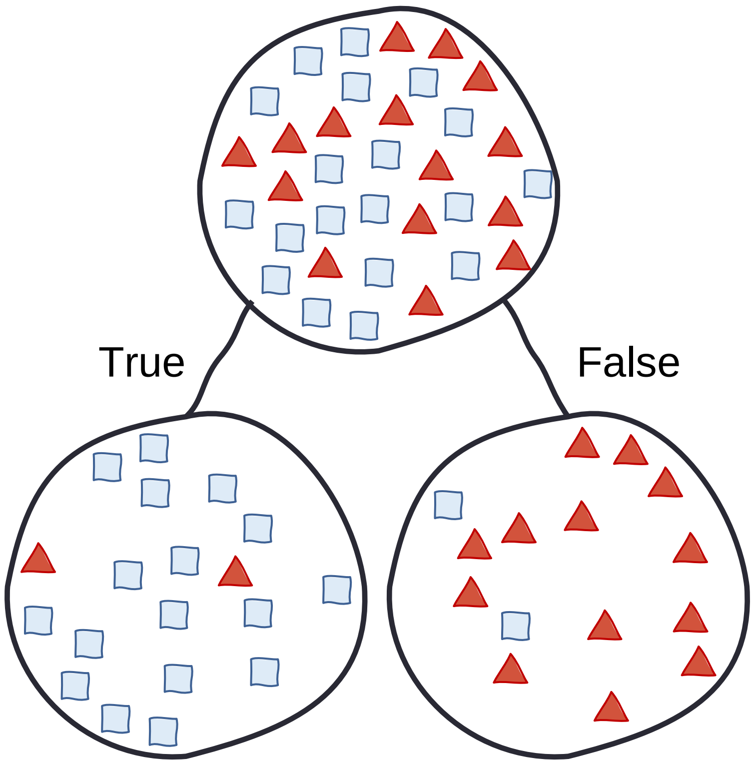


- Which split is preferable?
- Can we formalize it?

Feature: Likes apples?

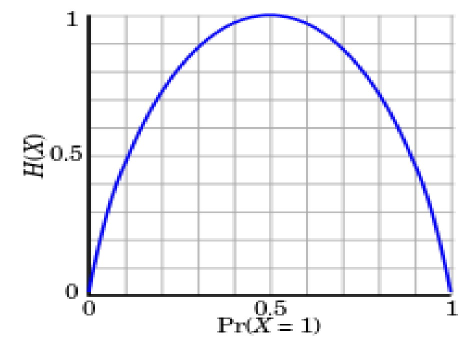


Feature: Likes ice cream?



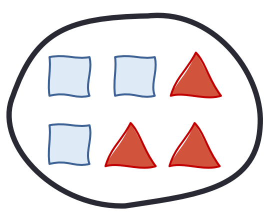
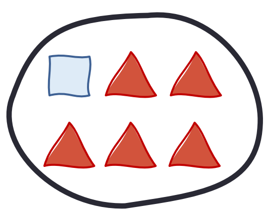
ID3

- Greedy TDIDT algorithm that uses entropy as an impurity measure.
- Entropy
  - An impurity measure.
  - For binary labels, the entropy in a tree node v is:



$$H(v) \triangleq -p_v \log_2 p_v - (1-p_v)\log_2(1-p_v)$$

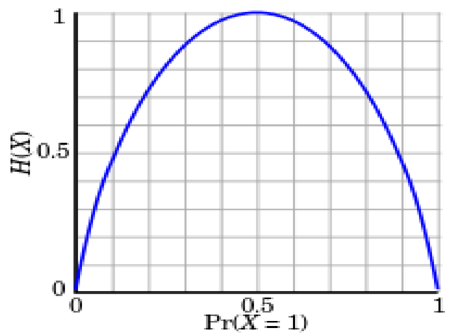
where $p_v \triangleq \frac{\left|\{(\boldsymbol{x}, y) \in v \mid y = 1\}\right|}{|v|}$.

- **Q:** $H\Bigg($  $\Bigg) = H(0.5) = ?$
- **Q:** $H\Bigg($  $\Bigg) = H(0.167) = ?$

ID3

- Greedy TDIDT algorithm that uses entropy as an impurity measure.

- Entropy: $H(v) \triangleq -p_v \log_2 p_v - (1 - p_v) \log_2(1 - p_v)$



- Splitting criterion: Information gain
  - Split by the feature a that reduces entropy the most.

entropy before split

$$\mathrm{IG}(v, a) = H(v) - \frac{|v_{a=T}|}{|v|} H(v_{a=T}) - \frac{|v_{a=F}|}{|v|} H(v_{a=F})$$

entropy after split (when $a = T$)

entropy after split (when $a = F$)

where $v_{a=T} \triangleq \{(\boldsymbol{x}, y) \in v | x_a = T\}$.

  - Goal at each iteration: find the feature with the largest information gain.

ID3: Dry run

- Let us demonstrate ID3 using a (made up) COVID-19 dataset.

ID	Fever	Cough	Smell loss	Corona
1	F	T	F	**F**
2	F	T	F	**F**
3	F	T	T	**T**
4	T	F	F	**F**
5	T	F	T	**T**
6	T	T	T	**T**
7	T	T	F	**T**

ID3: Dry run

ID	Fever	Cough	Smell loss	Corona
1	F	T	F	F
2	F	T	F	F
3	F	T	T	T
4	T	F	F	F
5	T	F	T	T
6	T	T	T	T
7	T	T	F	T

- Find the feature with the largest information gain:

$$\underset{a}{\text{argmax}}\, \text{IG}(v = \{1,2,3,4,5,6,7\}, a) = \underset{a}{\text{argmax}} \left(H(v) - \frac{|v_{a=T}|}{|v|} H(v_{a=T}) - \frac{|v_{a=F}|}{|v|} H(v_{a=F}) \right)$$

Attribute	$\frac{\lvert v_{a=T}\rvert}{\lvert v\rvert}$	$\frac{\lvert v_{a=F}\rvert}{\lvert v\rvert}$	$H(v_{a=T})$	$H(v_{a=F})$	$\text{IG}(v,a) - H(v)$
Fever	$4/7$	$3/7$	$H(3/4)$	$H(1/3)$	$-\frac{4}{7}H(3/4) - \frac{3}{7}H(1/3)$
Cough					
Smell loss					

ID3: Dry run

1, 2, 3, 4, 5, 6, 7

ID	Fever	Cough	Smell loss	Corona
1	F	T	F	F
2	F	T	F	F
3	F	T	T	T
6	T	T	T	T
7	T	T	F	T
4	T	F	F	F
5	T	F	T	T

- Find the feature with the largest information gain:

$$\operatorname*{argmax}_a \mathrm{IG}(v = \{1,2,3,4,5,6,7\}, a) = \operatorname*{argmax}_a \left(H(v) - \frac{|v_{a=T}|}{|v|} H(v_{a=T}) - \frac{|v_{a=F}|}{|v|} H(v_{a=F}) \right)$$

Attribute	$\frac{\lvert v_{a=T}\rvert}{\lvert v\rvert}$	$\frac{\lvert v_{a=F}\rvert}{\lvert v\rvert}$	$H(v_{a=T})$	$H(v_{a=F})$	$\mathrm{IG}(v,a) - H(v)$
Fever	$4/7$	$3/7$	$H(3/4)$	$H(1/3)$	$-\frac{4}{7}H(3/4) - \frac{3}{7}H(1/3)$
Cough	$5/7$	$2/7$	$H(3/5)$	$H(1/2)$	$-\frac{5}{7}H(3/5) - \frac{2}{7}H(1/2)$
Smell loss					

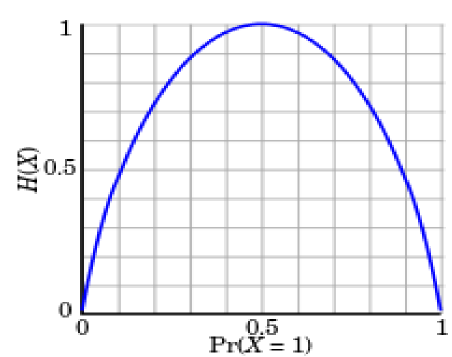
ID3: Dry run

ID	Fever	Cough	Smell loss	Corona
3	F	T	T	T
5	T	F	T	T
6	T	T	T	T
1	F	T	F	F
2	F	T	F	F
4	T	F	F	F
7	T	T	F	T

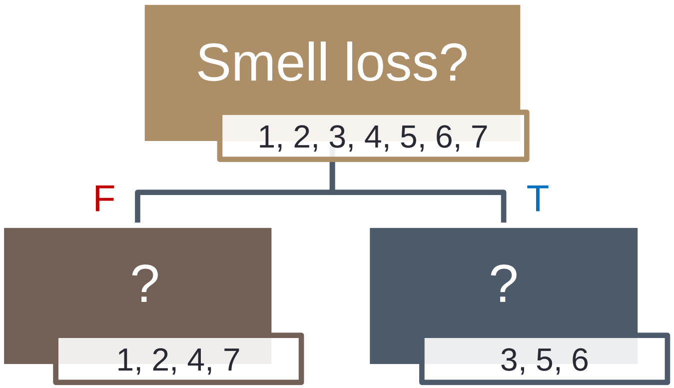
- Find the feature with the largest information gain:

$$\underset{a}{\text{argmax}}\, \text{IG}(v = \{1,2,3,4,5,6,7\}, a) = \underset{a}{\text{argmax}} \left(H(v) - \frac{|v_{a=T}|}{|v|} H(v_{a=T}) - \frac{|v_{a=F}|}{|v|} H(v_{a=F}) \right)$$

Attribute	$\frac{\lvert v_{a=T}\rvert}{\lvert v\rvert}$	$\frac{\lvert v_{a=F}\rvert}{\lvert v\rvert}$	$H(v_{a=T})$	$H(v_{a=F})$	$\text{IG}(v, a) - H(v)$
Fever	$4/7$	$3/7$	$H(3/4)$	$H(1/3)$	$-\frac{4}{7}H(3/4) - \frac{3}{7}H(1/3)$
Cough	$5/7$	$2/7$	$H(3/5)$	$H(1/2)$	$-\frac{5}{7}H(3/5) - \frac{2}{7}H(1/2)$
Smell loss	$3/7$	$4/7$	0	$H(1/4)$	$-\frac{4}{7}H(1/4)$

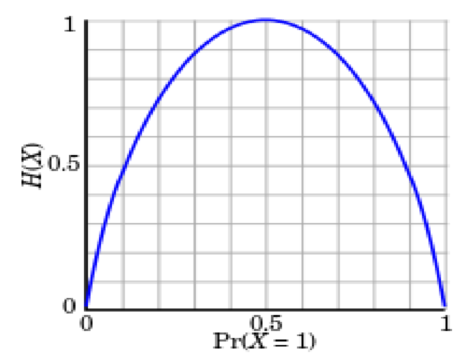


ID3: Dry run



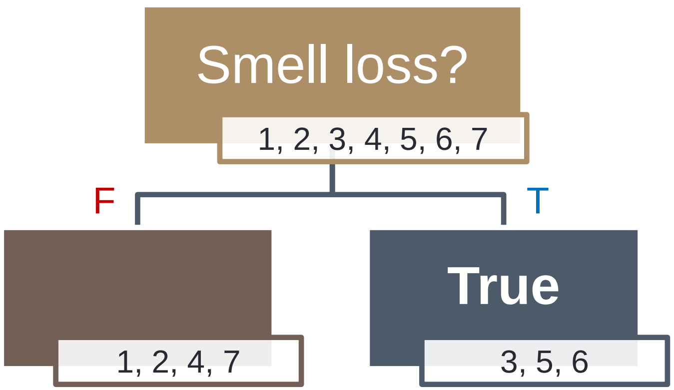
ID	Fever	Cough	Smell loss	Corona
3	F	T	T	T
5	T	F	T	T
6	T	T	T	T
1	F	T	F	F
2	F	T	F	F
4	T	F	F	F
7	T	T	F	T

Attribute	$\frac{\lvert v_{a=T}\rvert}{\lvert v\rvert}$	$\frac{\lvert v_{a=F}\rvert}{\lvert v\rvert}$	$H(v_{a=T})$	$H(v_{a=F})$	$\mathrm{IG}(v,a) - H(v)$
Fever	$4/7$	$3/7$	$H(3/4)$	$H(1/3)$	$-\frac{4}{7}H(3/4) - \frac{3}{7}H(1/3)$
Cough	$5/7$	$2/7$	$H(3/5)$	$H(1/2)$	$-\frac{5}{7}H(3/5) - \frac{2}{7}H(1/2)$
Smell loss	$3/7$	$4/7$	0	$H(1/4)$	$-\frac{4}{7}H(1/4)$



ID3: Dry run

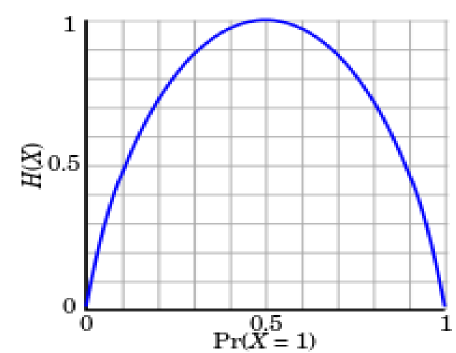
ID	Fever	Cough	Smell loss	Corona
1	F	T	F	**F**
2	F	T	F	**F**
4	T	F	F	**F**
7	T	T	F	**T**



- Find the feature with the largest information gain:

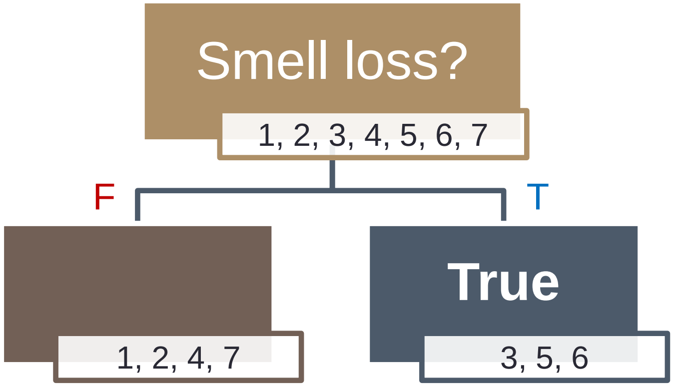
$$\underset{a}{\operatorname{argmax}}\, \mathrm{IG}(v = \{1,2,4,7\}, a) = \underset{a}{\operatorname{argmax}} \left(H(v) - \frac{|v_{a=T}|}{|v|} H(v_{a=T}) - \frac{|v_{a=F}|}{|v|} H(v_{a=F}) \right)$$

Attribute	$\frac{\lvert v_{a=T}\rvert}{\lvert v\rvert}$	$\frac{\lvert v_{a=F}\rvert}{\lvert v\rvert}$	$H(v_{a=T})$	$H(v_{a=F})$	$\mathrm{IG}(v,a) - H(v)$
Fever	$1/2$	$1/2$	$H(1/2)$	0	$-\frac{1}{2} H\left(\frac{1}{2}\right) = -\frac{1}{2}$
Cough					



ID3: Dry run

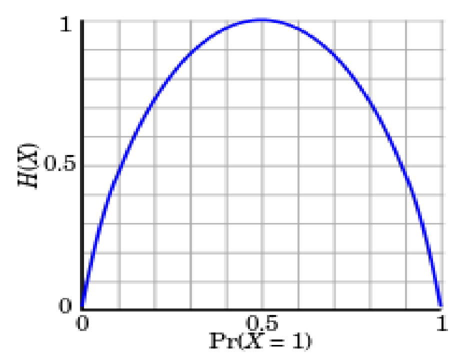
ID	Fever	Cough	Smell loss	Corona
1	F	T	F	F
2	F	T	F	F
4	T	F	F	F
7	T	T	F	T



- Find the feature with the largest information gain:

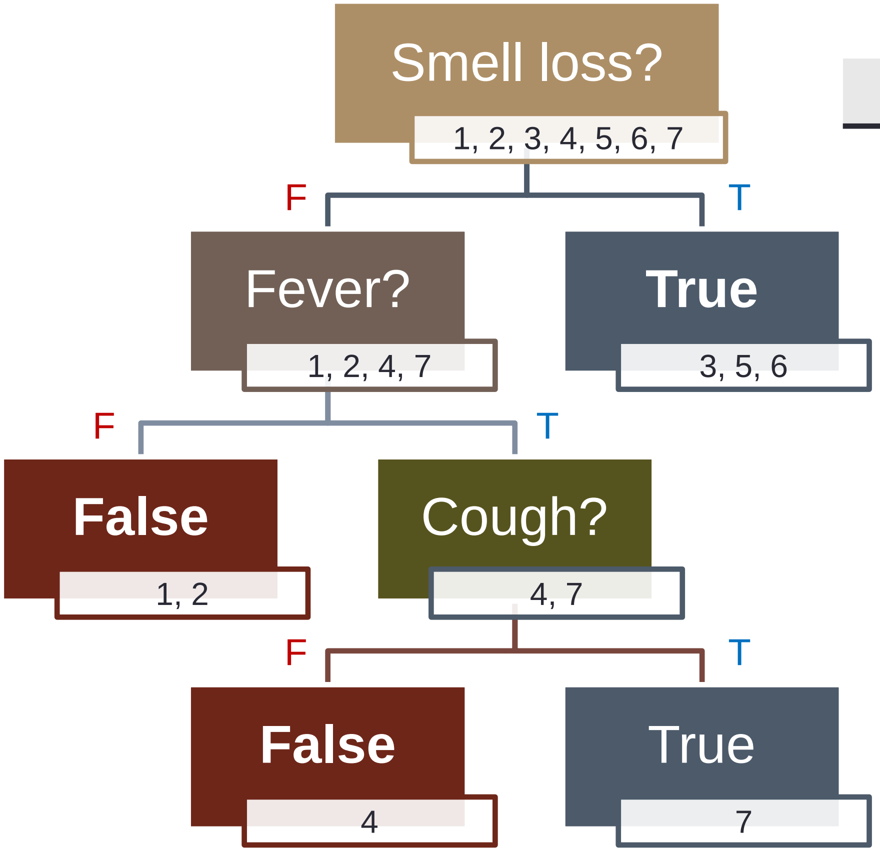
$$\operatorname*{argmax}_a \mathrm{IG}(v = \{1,2,4,7\}, a) = \operatorname*{argmax}_a \left(H(v) - \frac{|v_{a=T}|}{|v|} H(v_{a=T}) - \frac{|v_{a=F}|}{|v|} H(v_{a=F}) \right)$$

Attribute	$\frac{\|v_{a=T}\|}{\|v\|}$	$\frac{\|v_{a=F}\|}{\|v\|}$	$H(v_{a=T})$	$H(v_{a=F})$	$\mathrm{IG}(v,a) - H(v)$
Fever	$1/2$	$1/2$	$H(1/2)$	0	$-\frac{1}{2} H\left(\frac{1}{2}\right) = -\frac{1}{2}$
Cough	$3/4$	$1/4$	$H(1/3)$	0	$-\frac{3}{4} H\left(\frac{1}{3}\right) \approx -0.689$

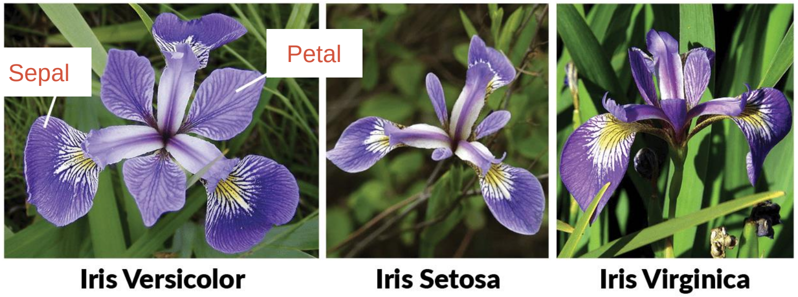


ID3: Dry run

ID	Fever	Cough	Smell loss	Corona
1	F	T	F	**F**
2	F	T	F	**F**
3	F	T	T	**T**
4	T	F	F	**F**
5	T	F	T	**T**
6	T	T	T	**T**
7	T	T	F	**T**



Iris dataset

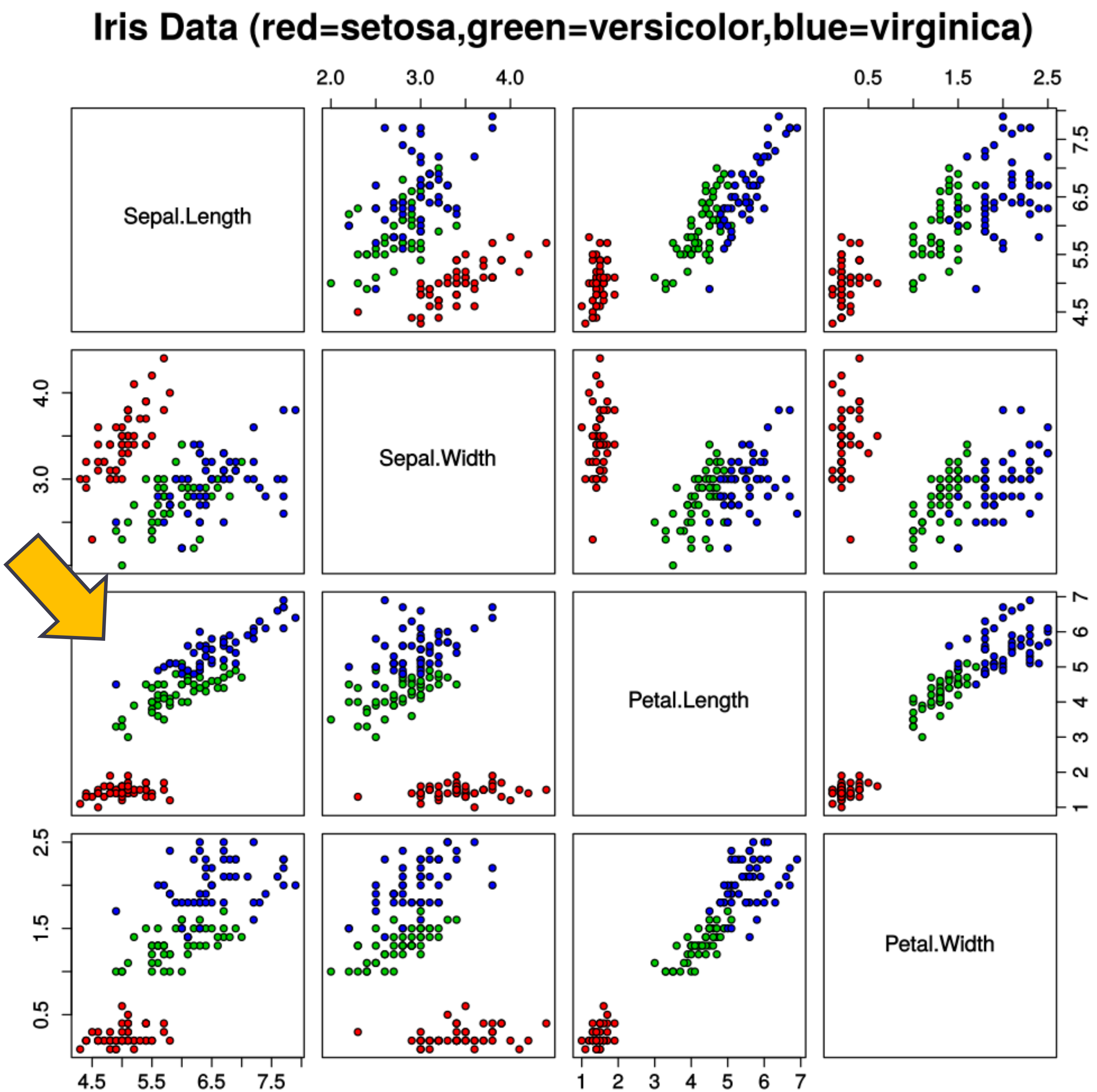


Source: ML in R

- From Wikipedia:
  - *A multivariate dataset introduced by the British statistician Fisher (1936).*
- Three classes:
  - Setosa, Versicolor, and Virginica
- Four features:
  - Sepal length, sepal width, petal length, petal width (in cm)
  - Sepal is עלה גביע
  - Petal is עלה כותרת

Iris dataset

- Let's choose 2 features for classification.
- Quick bivariate analysis (see sns.pairplot).
- Which seem good?



Source: Wikipedia

Iris dataset

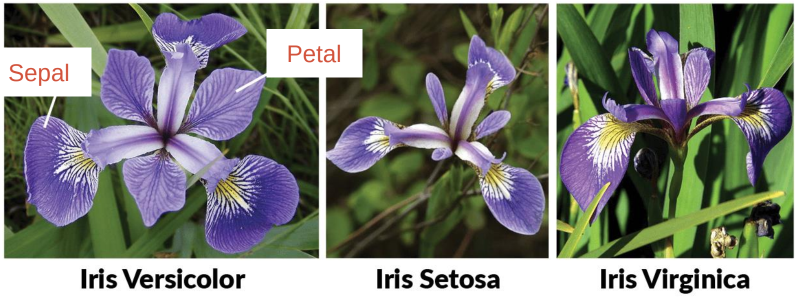
- Load the data

```
from sklearn.datasets import load_iris

iris = load_iris()
print(iris.data.shape)
print(iris.target.shape)
```

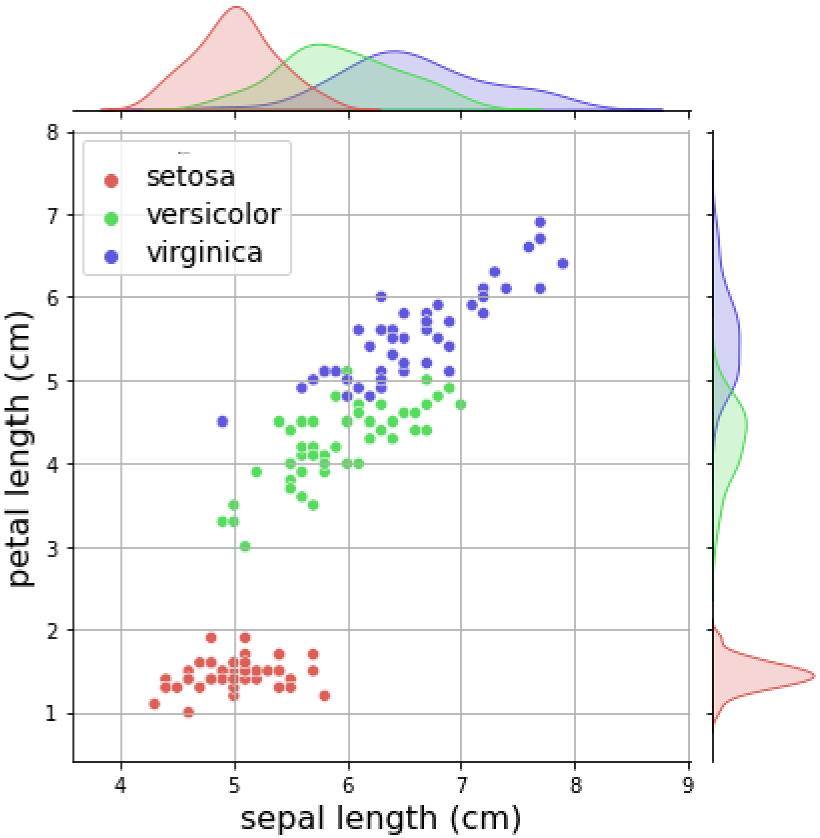
```
(150, 4) (150,)
```

- Visualize it

```
sns.jointplot(x=iris.data[:, 0],
              y=iris.data[:, 2],
              hue=iris.target)
```



Source: ML in R



Iris dataset

Iris Versicolor | Iris Setosa | Iris Virginica

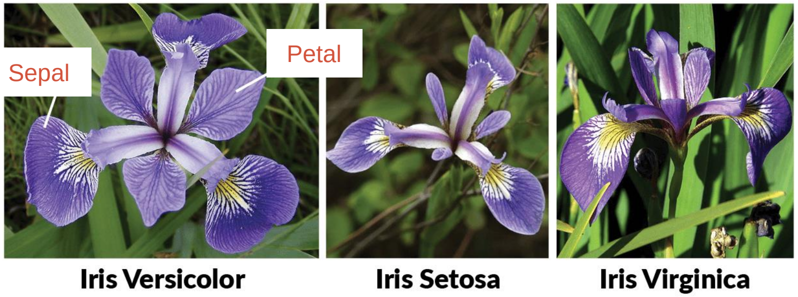
Source: ML in R

- Migrate to `pandas` (for convenience)

```
df = pd.DataFrame(
        data=np.c_[iris['data'], iris['target']],
        columns=iris['feature_names'] + ['target'])

df.sample(5)
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
93	5.0	2.3	3.3	1.0	1.0
122	7.7	2.8	6.7	2.0	2.0
39	5.1	3.4	1.5	0.2	0.0
141	6.9	3.1	5.1	2.3	2.0
61	5.9	3.0	4.2	1.5	1.0

Train-test split



Source: ML in R

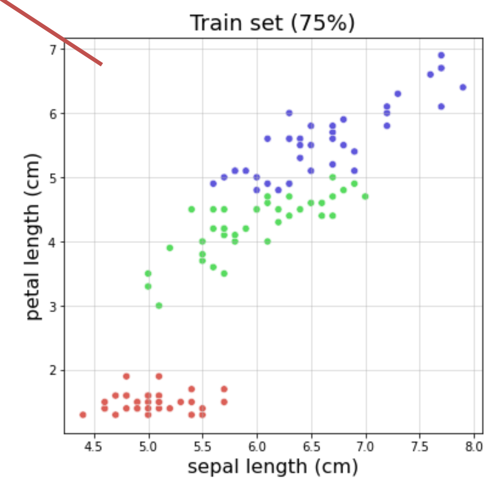
- Prepare for learning: split the data

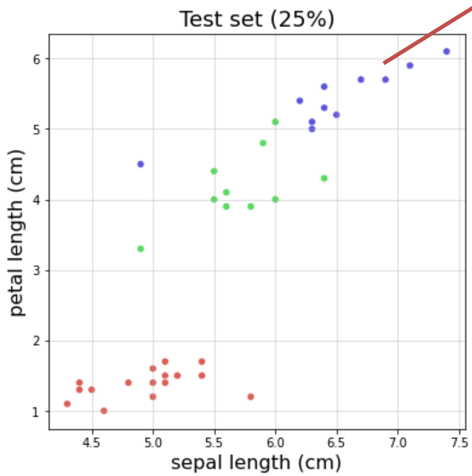
```
from sklearn.model_selection import train_test_split
train, test = train_test_split(df, test_size=0.25)

print(train.shape, test.shape)
```

```
(112, 5) (38, 5)
```

For training the classifier

For evaluating performance





Note: for such small datasets, there are better ways to evaluate performance (week 06).

Decision trees in sklearn

sklearn.tree.DecisionTreeClassifier

class `sklearn.tree.DecisionTreeClassifier`(*, *criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, class_weight=None, ccp_alpha=0.0*) [source]

A decision tree classifier.

Read more in the User Guide.

Parameters:

criterion : *{"gini", "entropy"}, default="gini"*

The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain.

splitter : *{"best", "random"}, default="best"*

The strategy used to choose the split at each node. Supported strategies are "best" to choose the best split and "random" to choose the best random split.

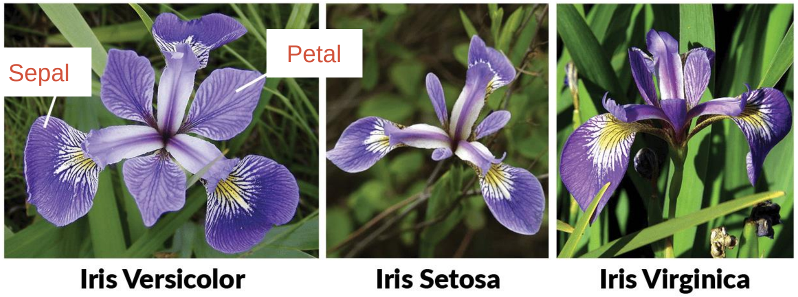
max_depth : *int, default=None*

The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

min_samples_split : *int or float, default=2*

The minimum number of samples required to split an internal node:

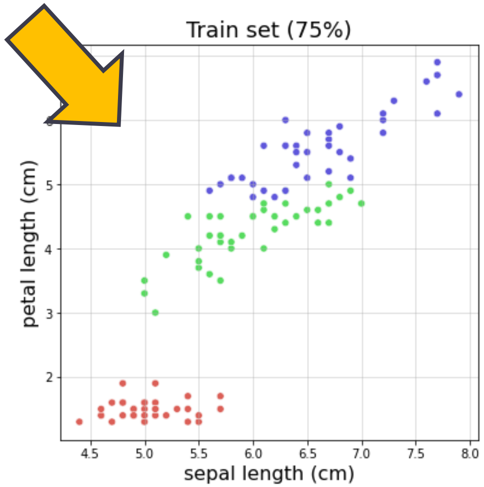
ID3 with sklearn

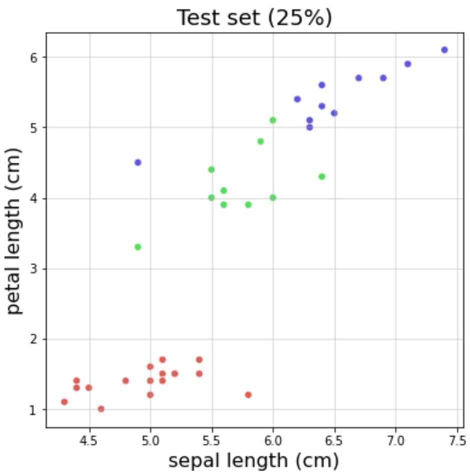


Source: ML in R

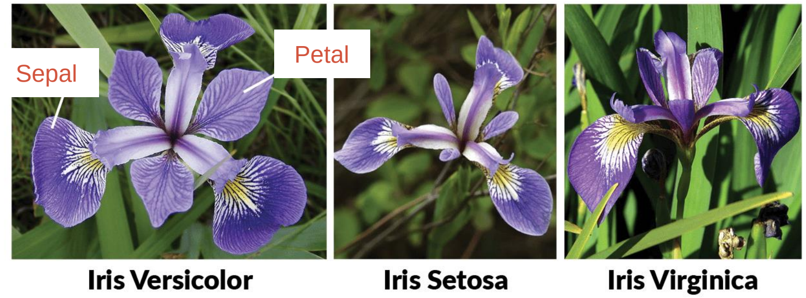
- Finally, we learn!

```
X_train = train.iloc[:, [0, 2]]
y_train = train["target"]

X_test = test.iloc[:, [0, 2]]
y_test = test["target"]

h = DecisionTreeClassifier(criterion="entropy")
h.fit(X_train, y_train)
```





ID3 with sklearn



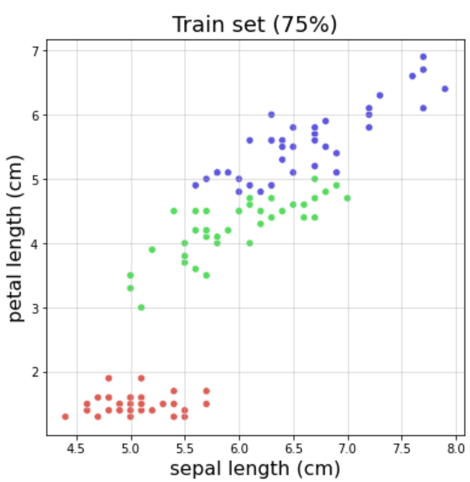
Source: [ML in R]

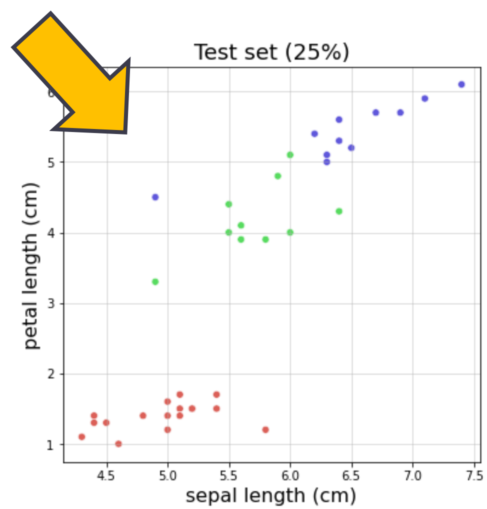
- We evaluate the trained model

```
trainAcc = np.sum(h.predict(X_train) == y_train) * 100 / len(y_train)
testAcc = np.sum(h.predict(X_test) == y_test) * 100 / len(y_test)
print("Train accuracy: {:.1f}%, Test accuracy: {:.1f}%".format(
         trainAcc, testAcc))
```

```
Train accuracy: 99.1%, Test accuracy: 89.5%
```

Q: Why not 100%?





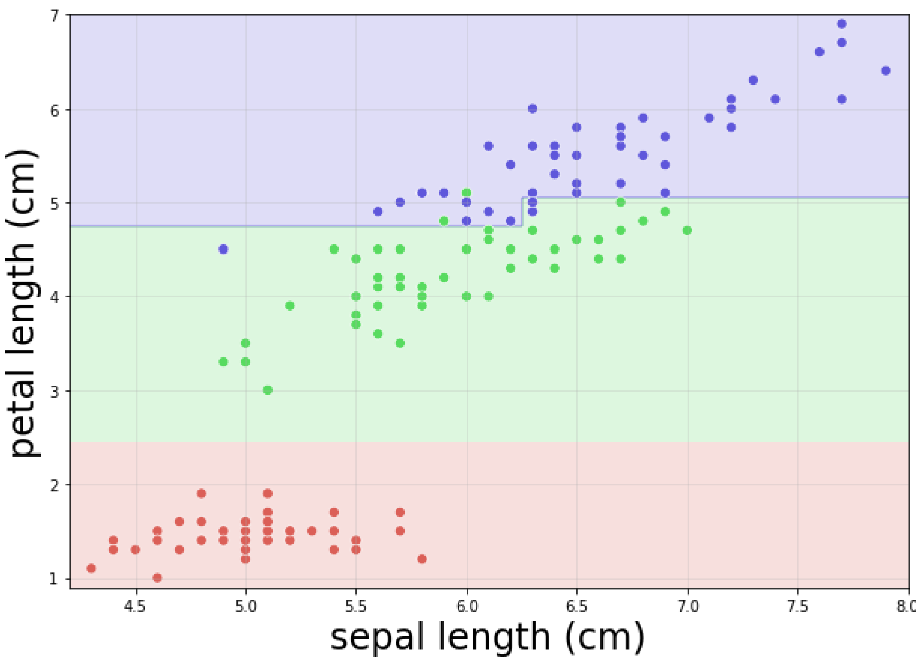
ID3 with sklearn

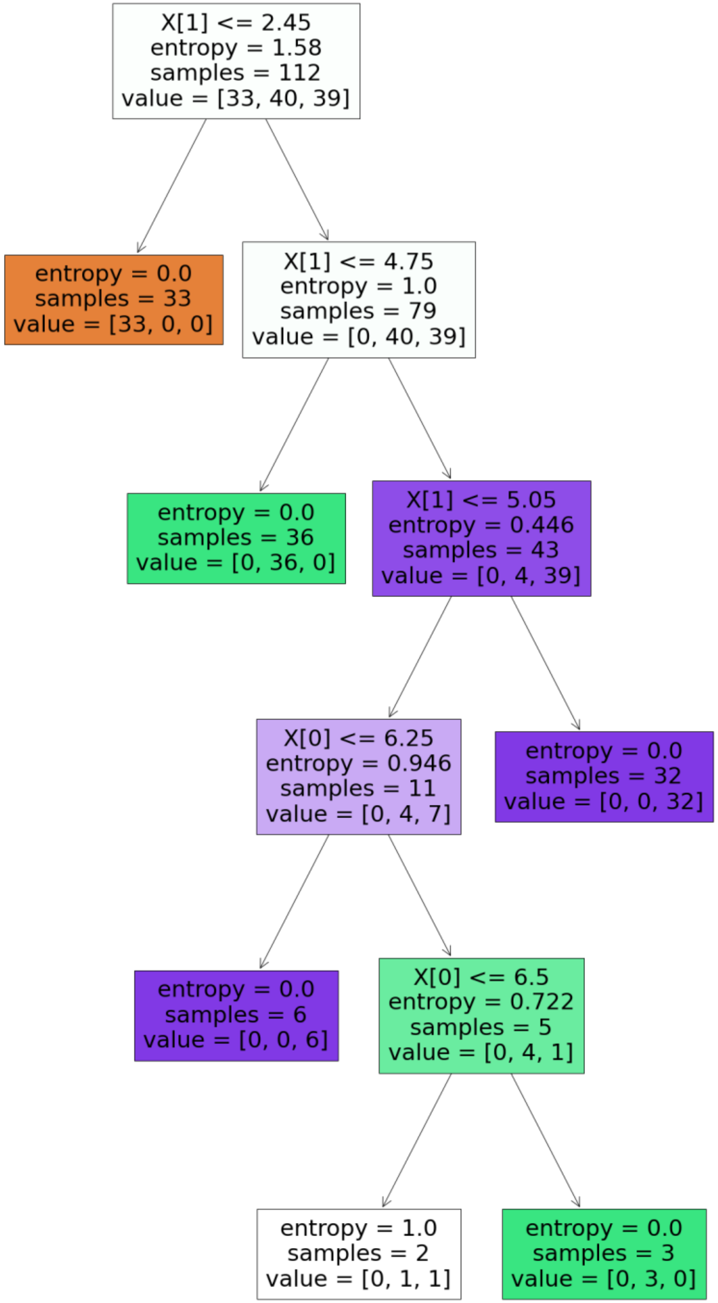
- Again, visualize!

```python
from sklearn.tree import plot_tree
plot_tree(h, filled=True)
```

- Notice the high interpretability.

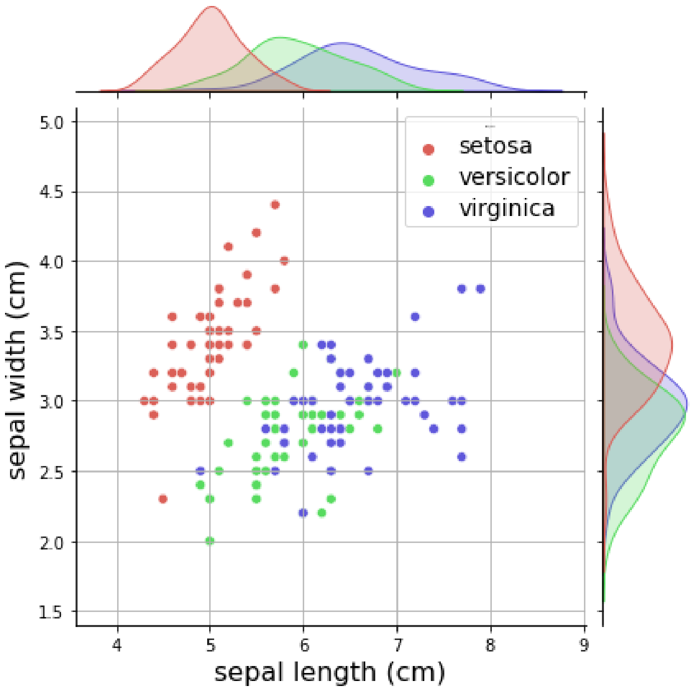
- We can also visualize the decision regions.



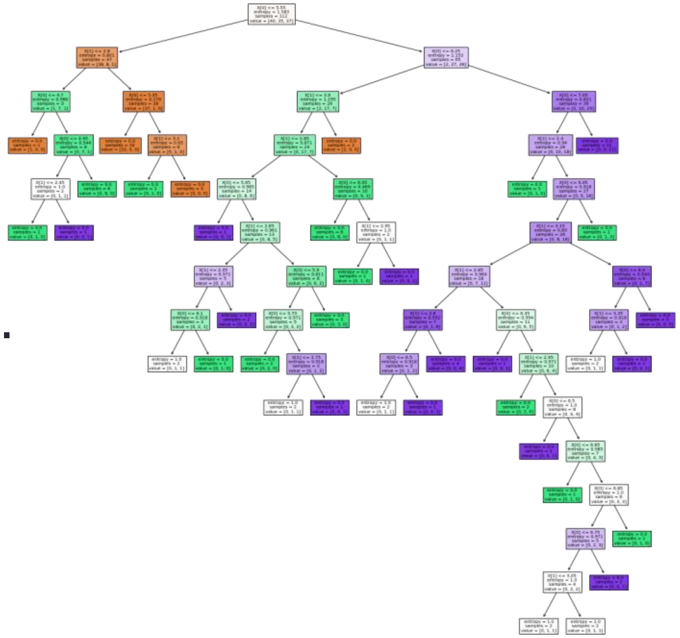


Overfitting

- The algorithm builds a tree until all samples are correctly classified.
- Is that necessarily a good thing?
  - Demonstrate with two other, less separable, features.



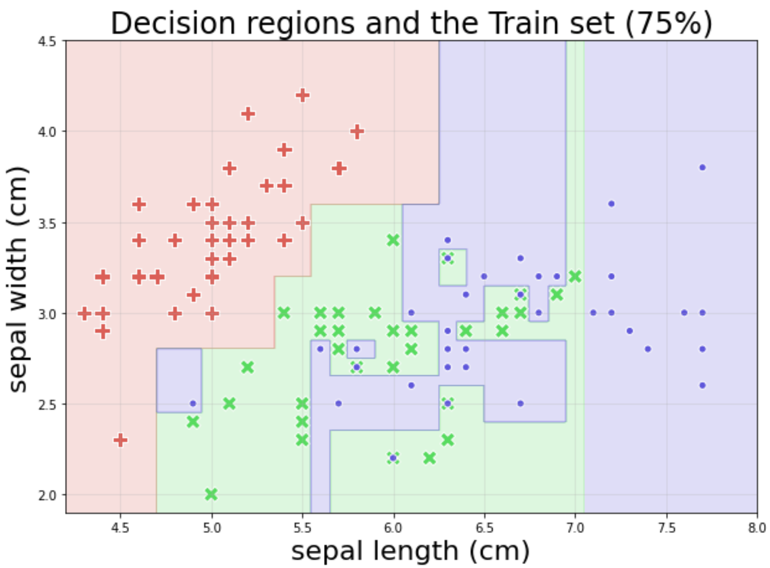
Overfitting

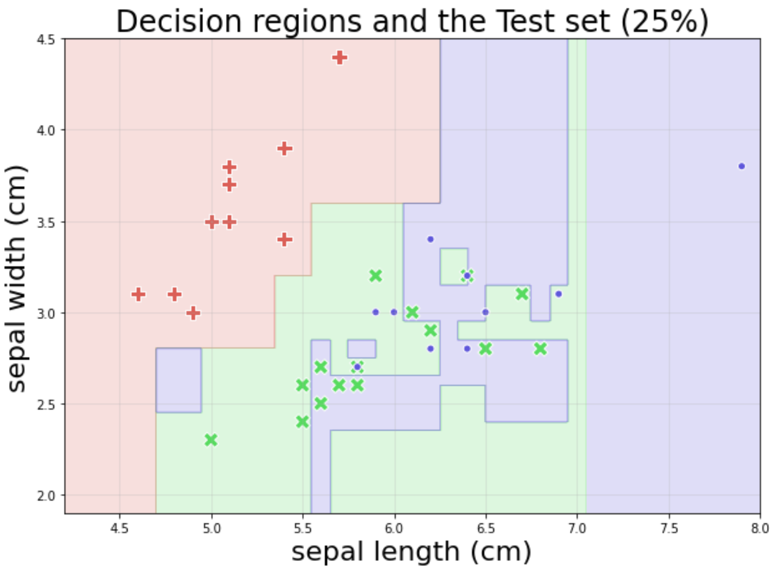


- The tree (almost) perfectly fits the training set.

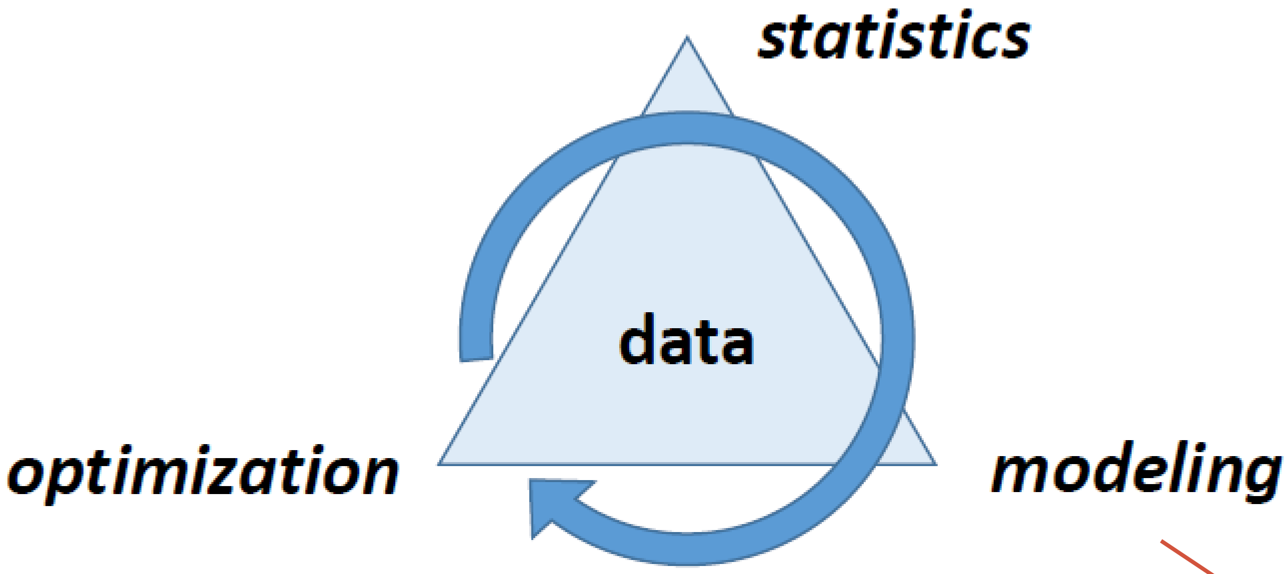
```
Train accuracy: 94.6%, Test accuracy: 63.2%
```

- But the decision rule is overcomplicated and does not generalize well.





Three pillars of learning



The model class is too rich!

Controlling model complexity

- **Q:** Suggest ways to restrict greedy TDIDT algorithms from building overcomplicated models.

- **A:** Use stopping criteria!
  - Limit depth of tree;
  - Minimum number of samples in nodes;
  - And more…

Parameters:

max_depth : *int, default=None*

The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

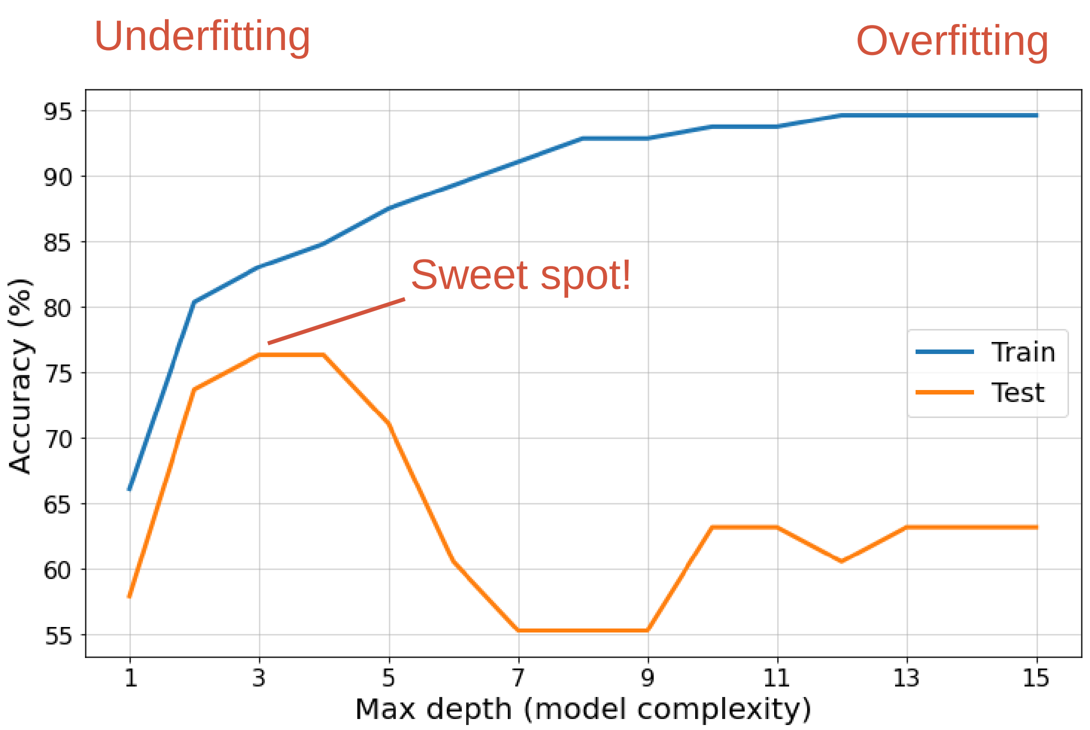
min_samples_split : *int or float, default=2*

The minimum number of samples required to split an internal node:

- If int, then consider `min_samples_split` as the minimum number.

Controlling model complexity

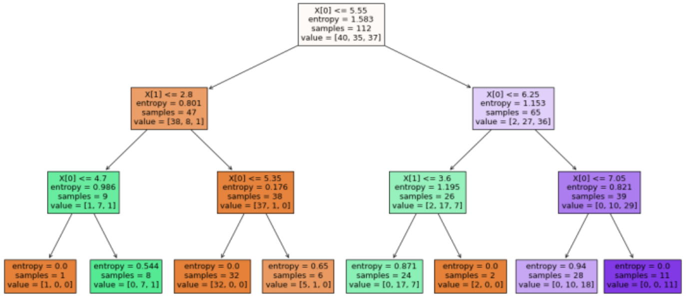
```
for depth in range(1, 16):
  h = DecisionTreeClassifier(criterion="entropy", max_depth=depth)
  h.fit(X_train, y_train)
```

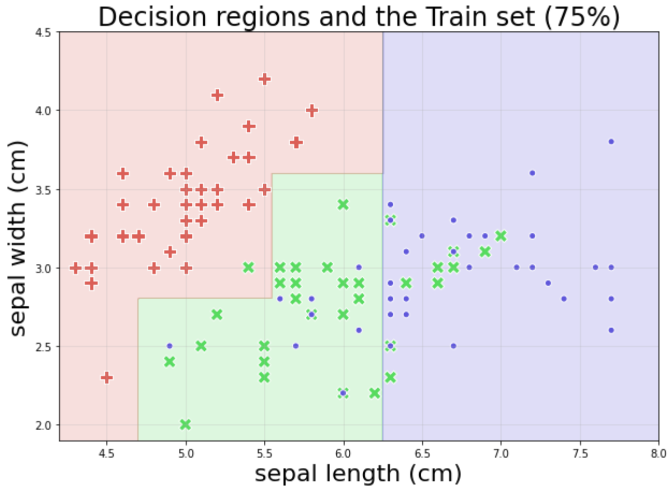


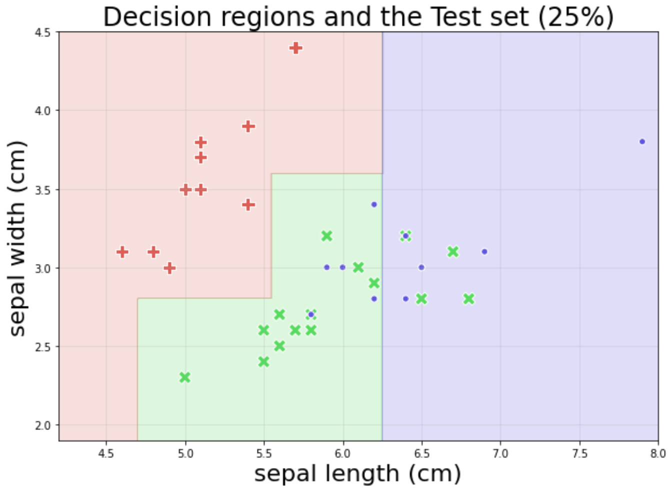
Controlling model complexity

```
h = DecisionTreeClassifier(criterion="entropy", max_depth=3)
h.fit(X_train, y_train)
```

- Tree doesn't perfectly fit the training set, but decision rule is much simpler







Summary

- Decision trees are simple and interpretable models
- Optimization by greedy algorithms
  - Splitting criteria.
    - ID3 uses entropy.
- Prone to overfitting
  - Stopping criteria