A DECISION TREE ‘IS IT BIGGER THAN A BREADBOX’?

ROBERT BOBKOSKIE

Before We Dive In: Machine Learning Level Set

What is Machine Learning?

Interchangeable names: [*dataset = problem*], [*class = target = label*], [*features, attributes, dimensions*], [*samples = instances = observations*], **…**

- Machine Learning[1] (ML) gives computers the ability to learn without being explicitly programmed (Arthur Samuel, 1959). ML evolved from the study of pattern recognition and computational learning theory in artificial intelligence. ML explores the study and construction of algorithms that can learn from, and make predictions on data. Such algorithms transcend the paradigm of following strictly static program instructions by making data driven predictions or decisions through building a model from sample inputs.
  - Supervised Learning[2] infers a function from labeled training data. The training data consists of a set of training examples. In supervised learning, each example (instance) is a pair consisting of an input object of features (typically a vector) and a (labeled) output value: Decision Tree, KNN, SVM, Neural Networks.
    - ✓ **Classification**: The problem consists of two (or more, ***multi-class***) classes, and the attributes are classified (output) as one (or more, ***multi-label***) discrete valued labels.
    - ✓ **Regression**: The output is continuous rather than discrete.
  - Unsupervised Learning[3] Infers a function to describe hidden structure from "unlabeled" data (a classification or categorization is not included in the observations). Since the examples given to the learner are unlabeled, there is no objective evaluation of the accuracy of the structure that is output by the relevant algorithm: Clustering (k-means), Anomaly Detection, Neural Networks (Hebbian Learning).
- **Business Case**: Behavioral based detection in network security. Use a combination of supervised and unsupervised machine learning models to identify patterns and anomalies in big data.

Is it bigger than a Breadbox?

The question provides some insight into the definition of a good question. Asking good questions is intuitive for humans, but not for machines.

- Recursive partitioning: Spilt until label in feature space is pure (leaf node); else split on max information gain.
  - **Goal**: achieve perfect classification (or prediction) with minimal number of decisions. This is not always possible due to noise or inconsistencies in the data. Trying too hard leads to overfitting.
  - **Classification Tree**: Recursively partition (split) the data space by measuring impurity (Entropy, Gini) in the split to classify discrete valued labels.
  - **Regression Tree**: Measure statistical impurity [Residual Sum of Squares, Sum of Squared Error) to predict 'ordered' continuous valued output.

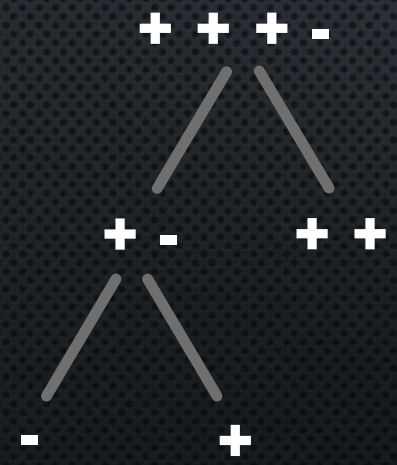
- A decision tree has nodes and leaves. The nodes ask the 'questions', the leaves are the 'predicted' labels.
  - **Inductive bias <=> Occam's Razor**:
    - ✓ Less complex decision trees are preferred.
    - ✓ Trees that place attributes with high information gain towards the root are preferred.
    - ✓ **Occam's Razor**: A simpler hypothesis is preferred over more complex ones.
  - **Overfitting <=> Generalization**:
    - ✓ Model performs better (accuracy) on the training data than the test data.
    - ✓ More complex decision trees tend to overfit and do not generalize well to the test data.
    - ✓ Constrain (max-depth, splitting criteria), or prune (C4.5) to reduce overfitting.

Decision Tree **Classification**

The Problem:

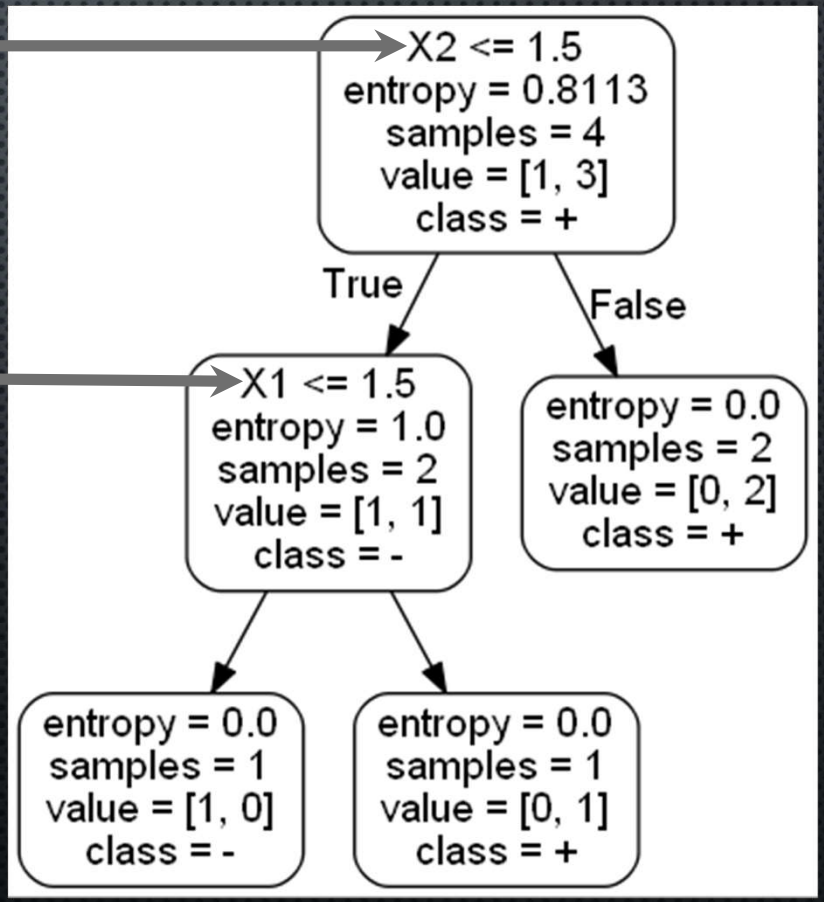
X = *[[1, 1], [1, 2], [2, 1], [2, 2]]*
Y = *[-1, 1, 1, 1]* → *multi-class,* → *multi-label*

- Two Features: [X_1, X_2]
- Two Labels: [**+**, **-**]
- Four Instances (composed of the feature matrix X_i and *Y* vector): [*three* **+**, *one* **-**]

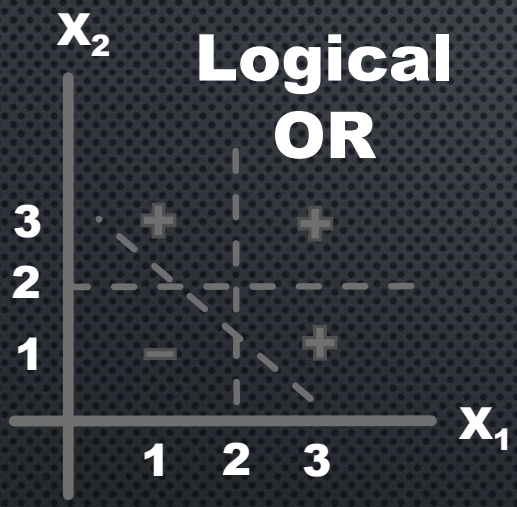


Note the choice for the decision boundary (***1.5***) does not maximize the margin. A Support Vector Machine would choose ***2***, thus maximizing the margin between the classes.

SKLEARN: Decision Tree



Splitting



Is the Data Linearly Separable? **YES**

Entropy: $\mathrm{H} = \sum_{i=1}^{n} -P_i log_2(P_i)$

First Split: H(Y) ~ 0.81

```
>>> import math
>>> -1.0/4*math.log(1.0/4, 2) \
... -3.0/4*math.log(3.0/4, 2)
0.8112781244591328
```

Conditional Entropy: H(Y|X)

$P(X_2 > 2) = P(X_2 < 2) = 2/4 = 1/2$

$H(Y|X_2 > 2) = 0$ *(labels are pure)*
$H(Y|X_2 < 2) = 1$

```
>>> -1.0/2*math.log(1.0/2, 2) \
... -1.0/2*math.log(1.0/2, 2)
1.0
```

Information Gain: H(parent) - [weighted ave]*H(children)

$IG(Y, X=X_2) = H(Y) - H(Y|X=X_2)$
$IG(Y, X=X_2) = 0.81 - (1/2)*1 - (1/2)*0$
$IG(Y, X=X_2) = 0.81 - 0.50 = 0.31$

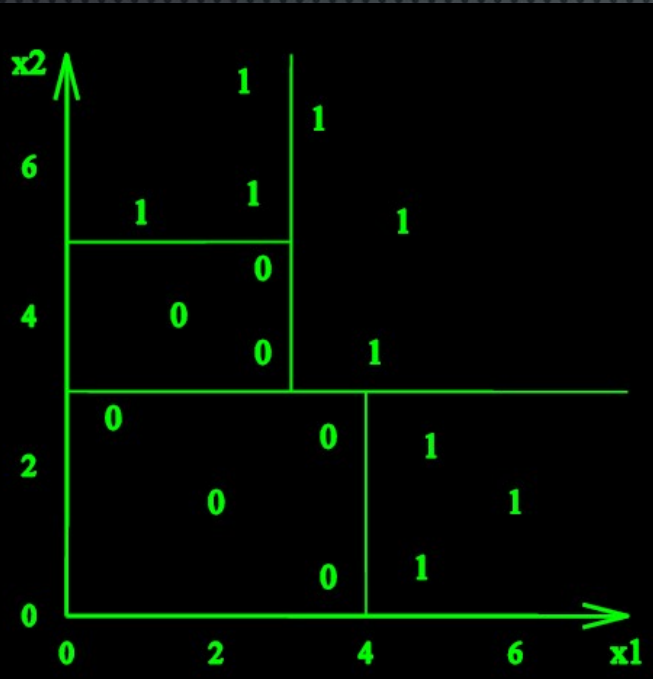
Second Split: H(Y) = 1.0

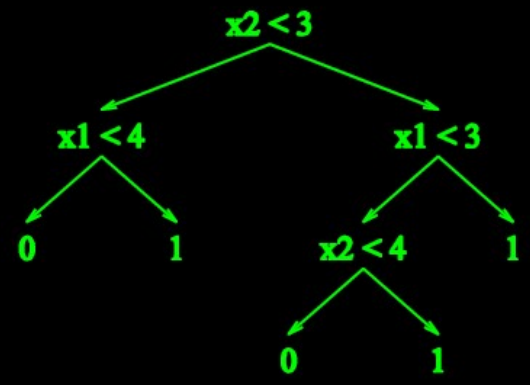
```
>>> -1.0/2*math.log(1.0/2, 2) \
... -1.0/2*math.log(1.0/2, 2)
1.0
```

$H(Y|X_1 > 2) = 0$ *(labels are pure)*
$H(Y|X_1 < 2) = 0$ *(labels are pure)*

$IG(Y, X=X_i) = 1.0 - 0 = 1.0$

More Splitting: Decision Boundaries





Entropy: H(Y)

```
>>> -9.0/16*math.log(9.0/16, 2) \
... -7.0/16*math.log(7.0/16, 2)
0.9886994082884974
```

Decision Boundaries:

- A decision tree is a nonlinear classifier, e.g., no requirement for data to be linearly separable.
- Decision trees divide the feature space into axis-parallel rectangles and label each rectangle with one of the K classes (labels).

Define a Problem: Start with Some Data

Preprocess Data: Pandas, Numpy

- The data may contain Strings and Null values that required modification before inputting to the Decision Tree Classifier.
- Use pandas to: map strings to ints, eliminate data with 'undetermined' or 'Null' values.
- Use Numpy slicing to partition the data set into [X, y], where one column is the target class [y] and all other columns are the feature instances [X].

For this demo: two problems (data sets), maybe more:

- The IRIS data set Is comprised of 3 classes of 50 instances, where each class refers to a type of iris. There are 4 attributes (features): sepal length, sepal width, petal length and petal width. The objective (problem) is to classify the species of iris based on the 4 attributes.
- A data set related with a direct marketing campaign of a Portuguese banking institution. The marketing campaign was initiated to sell a bank term deposit (product) to clients. The data consists of 4119 instances with 20 attributes of the client: [education, marital status, existing loans, day of week, etc.]. The objective is to classify (label), e.g., predict the outcome, 'client subscribed': ('yes') or ('no'). After preprocessing, 3811 instances, 18 attributes, 1905 instances each for test and training.
- Spam data: 4600 instances, 57 attributes (word counts and length of sequences of consecutive capital letters). The data was partitioned into 2300 instances each for test and training. The objective is to classify the e-mail as spam (1) or not (0).

Build a Tree: Cross Validation is Your Friend

Classify the **training data** and **generalize**, e.g., fit to the **test data**:

Evaluate hyper parameters and select the best ones

```
hyper_params = {"criterion": ["gini", "entropy"],
                "min_samples_split": [2, 10, 20, 100, 1000],
                "max_depth": [None, 2, 5, 10, 100],
                "min_samples_leaf": [1, 5, 10],
                "max_leaf_nodes": [None, 5, 10, 20]}
```

Use a 10-fold cross validation against the training data to evaluate best hyper-parameters

```
best_params = self.my_utils.Grid_Search(DecisionTreeClassifier(), name,
                                        X_train, y_train, hyper_params,
                                        options['num_folds'])
```

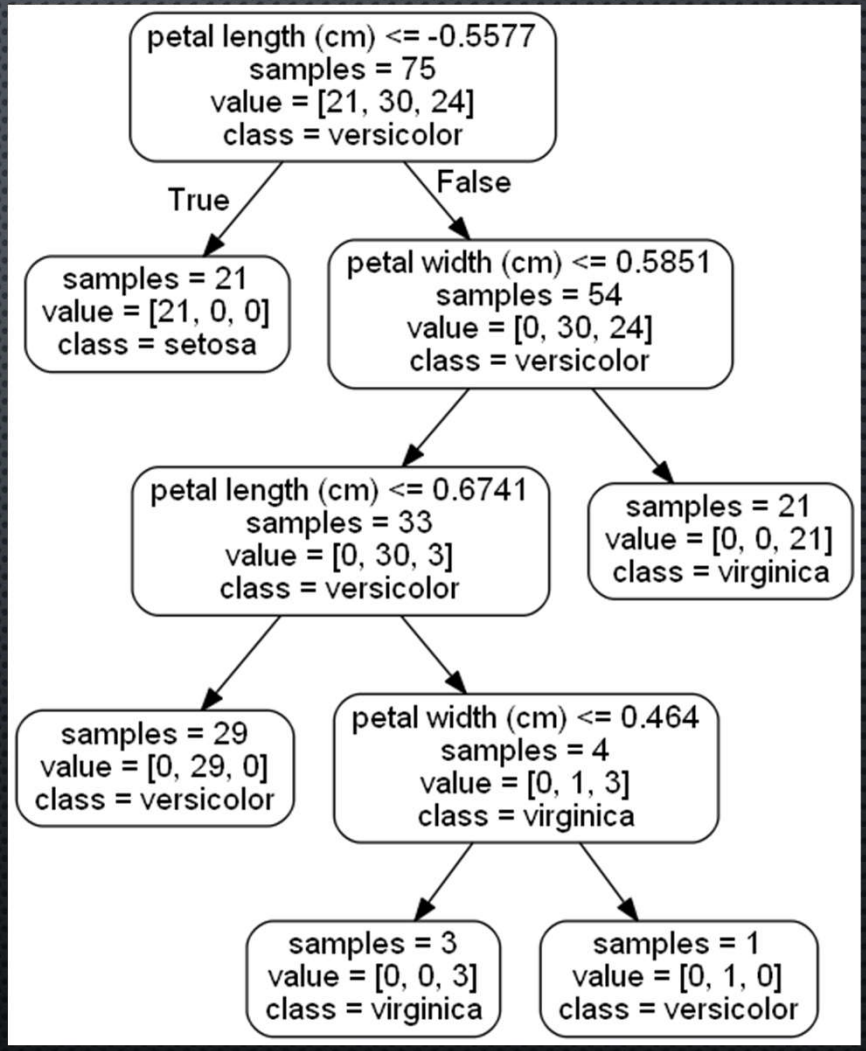
Classify and fit

```
d_tree = DecisionTreeClassifier(criterion=best_params['criterion'],
                                max_depth=best_params['max_depth'],
                                min_samples_split=best_params['min_samples_split'],
                                min_samples_leaf=best_params['min_samples_leaf'],
                                random_state=None,
                                max_leaf_nodes=best_params['max_leaf_nodes'])

d_tree.fit(X_test, y_test)
```

Plot the Decision Tree

- Petal length is the feature that provides the maximum information gain out of all the attributes.
- A leaf node, 21 instances can be classified as 'setosa' immediately from the root.
- In general, a decision tree with more depth is prone to overfitting. Normally, constraining (pre-pruning) the tree to control depth generalizes the test data with greater accuracy.
- Even better is to grow the tree to its maximal depth, then post-prune (C4.5). Unfortunately, not implemented in SKLEARN.



ASK GOOD QUESTIONS