An Introduction to Machine Learning

Alex Pizzuto

IceCube Bootcamp: June 2019
WIPAC / UW-Madison

GitHub repo with materials: https://github.com/apizzuto/bootcamp-machine-learning (https://github.com/apizzuto/bootcamp-machine-learning)

*Lots of material in these slides are from talks on ML by James Bourbeau or Sebastian Raschka

Outline

- What is Machine Learning?
  - Machine Learning vs. Classical Programming
  - Supervised vs. Unsupervised Machine Learning
- Data Representation
- Machine Learning Algorithms
  - Tree Based Learning
- Model Validation
- Machine Learning in IceCube

Machine Learning (ML)

"Machine Learning is the hot new thing"
— John L. Hennessy, Stanford President

Machine Learning (ML)

Slightly more seriously:
"Machine learning is the field of study that gives computers the ability to learn without being explicitly programmed"
— Arthur Samuel (Vanguard of AI)

"A machine-learning system is trained rather than explicitly programmed. It’s presented with many examples relevant to a task, and it finds statistical structure in these examples that eventually allows the system to come up with rules for automating the task."
— Francois Chollet, *Deep Learning with Python*

Classical Programming

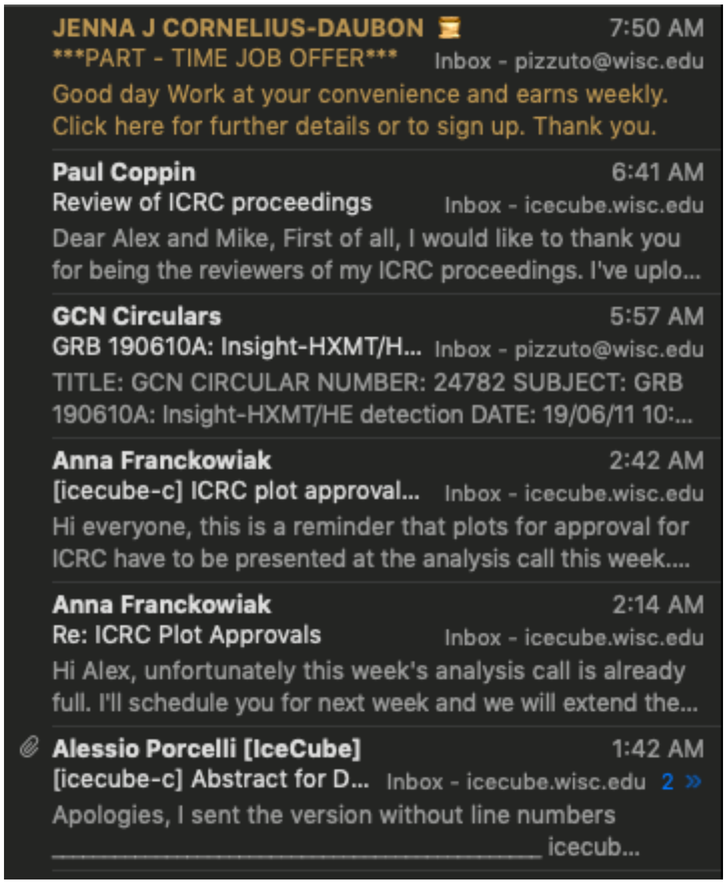
Suppose we want to write an algorithm to classify messages as `spam` or `not spam`. The classical approach:

In [13]:
```
def spam_filter(email):
    """Function that labels an email as 'spam' or 'not spam'
    """
    if 'Act now!' in email.contents:
        label = 'spam'
    elif 'hotmail.com' in email.sender:
        label = 'spam'
    elif email.contents.count('$') > 20:
        label = 'spam'
    else:
        label = 'not spam'

    return label
```

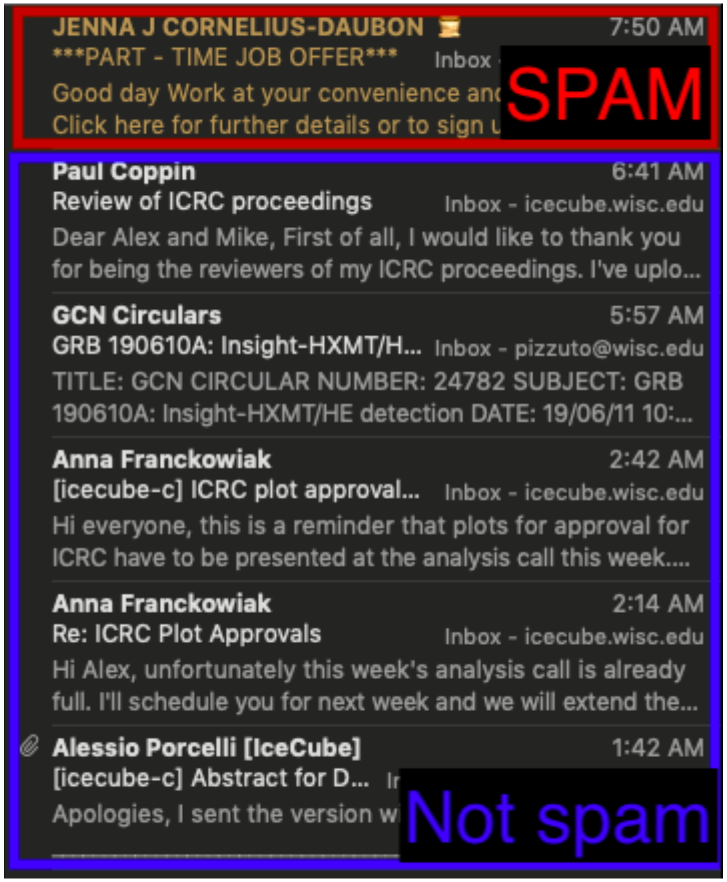
The Machine Learning Approach

Provide an example of some emails



The Machine Learning Approach

Provide an example of some emails
Label them as spam or not spam
Let the computer figure out the rules



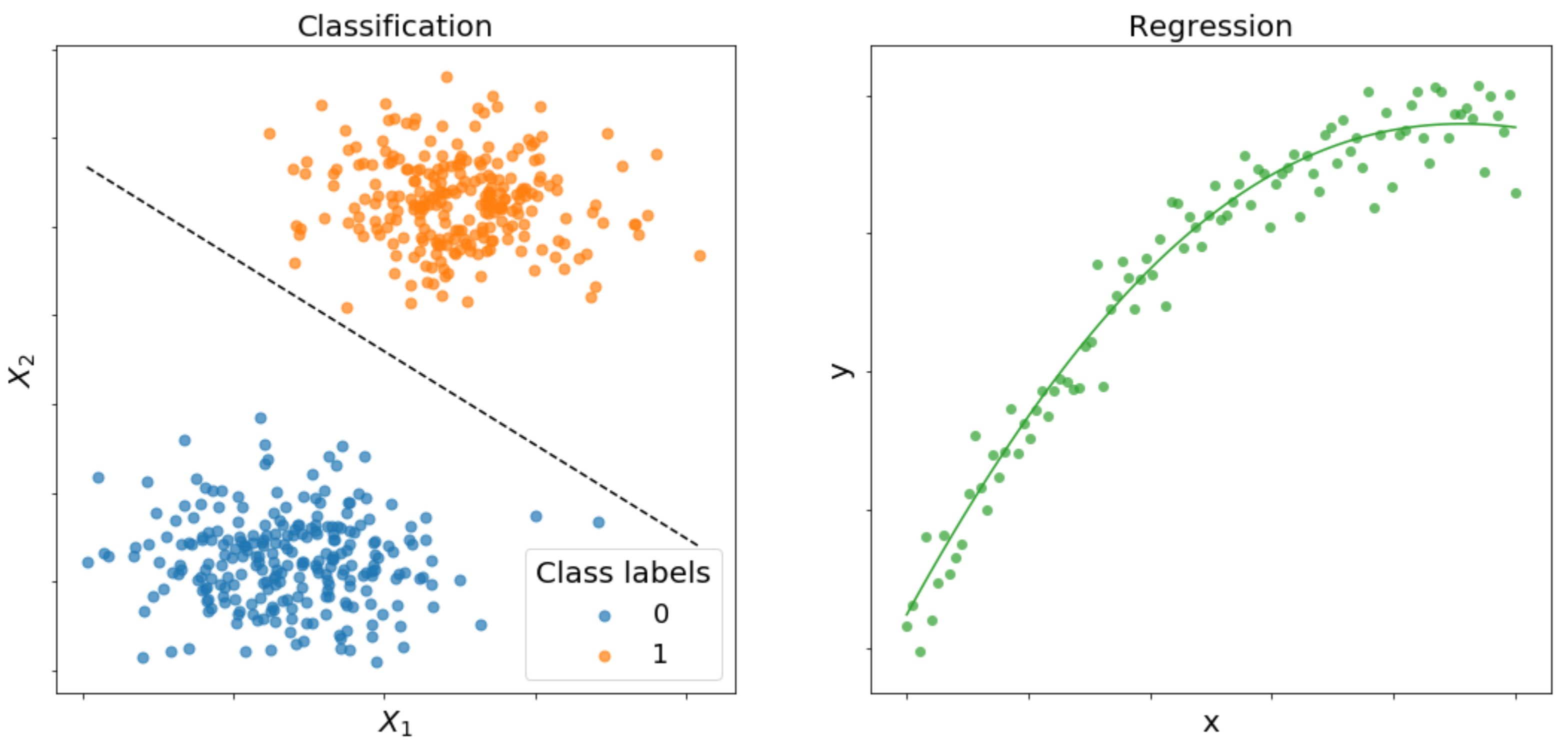
Types of ML

Supervised learning	Train a model using *labeled* training data in order to make prediction about future unseen data
Reinforcement learning	Agent maximizes some reward function via interacting with its environment
Unsupervised learning	Train a model using *unlabeled* training data in order to find underlying structure in data

Supervised Learning

Supervised learning is broken down into two categories, *Classification* and *Regression*

In [14]:
```
plotting.plot_classification_vs_regression()
```



Data Representation

We represent our data as a 2D array. Each row represents a *sample*, and each column represents a *feature*

Each sample has a correct classification, called a *target*, and we represent these as a column vector

sample (i.e. an event)

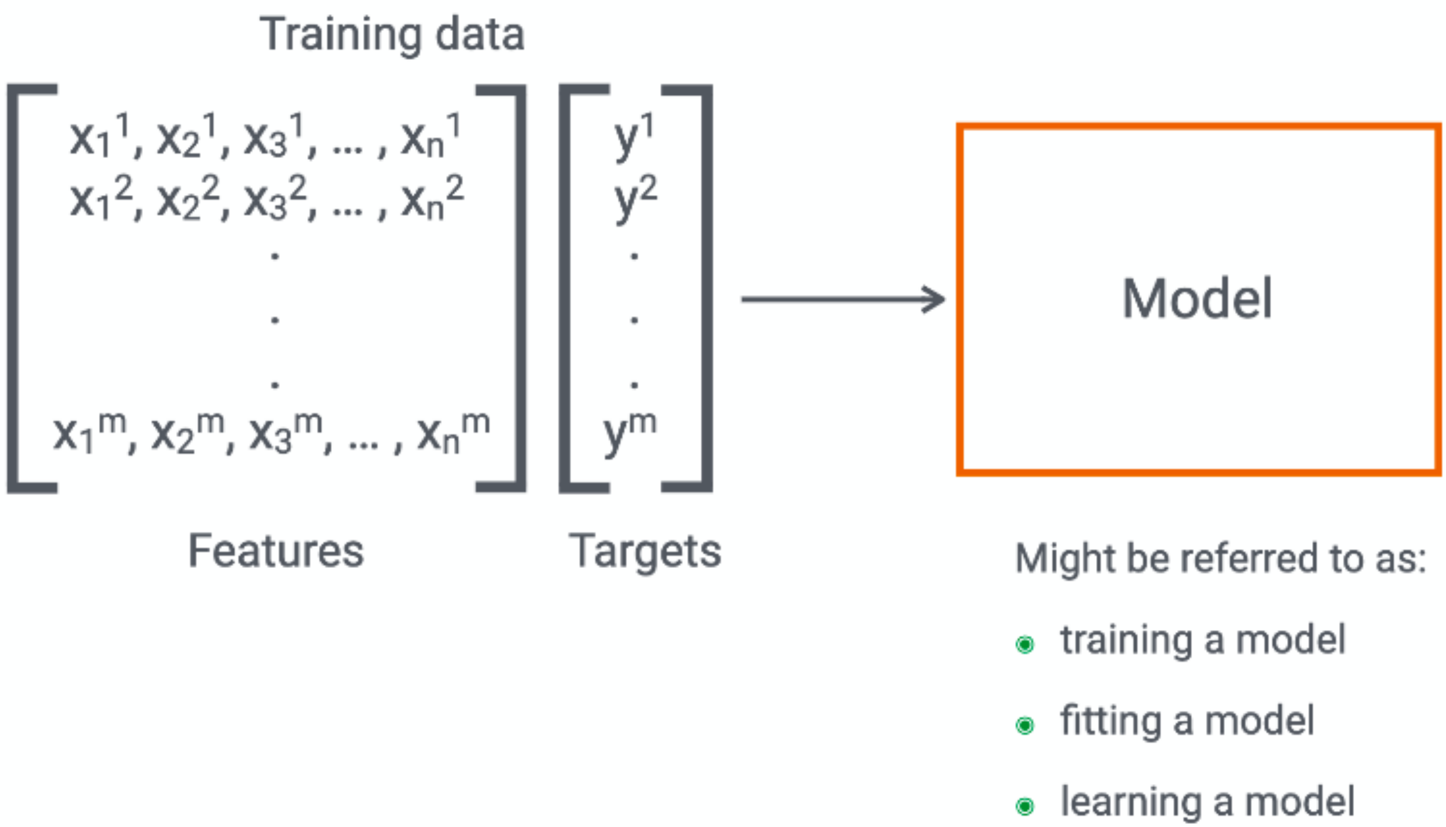
$$\mathbf{X} = \begin{bmatrix} x_1^{[1]} & x_2^{[1]} & \dots & x_m^{[1]} \\ x_1^{[2]} & x_2^{[2]} & \dots & x_m^{[2]} \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{[n]} & x_2^{[n]} & \dots & x_m^{[n]} \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} y^{[1]} \\ y^{[2]} \\ \vdots \\ y^{[n]} \end{bmatrix}$$

feature (energy, track length, etc.)

One true label for each sample (i.e. track or cascade)

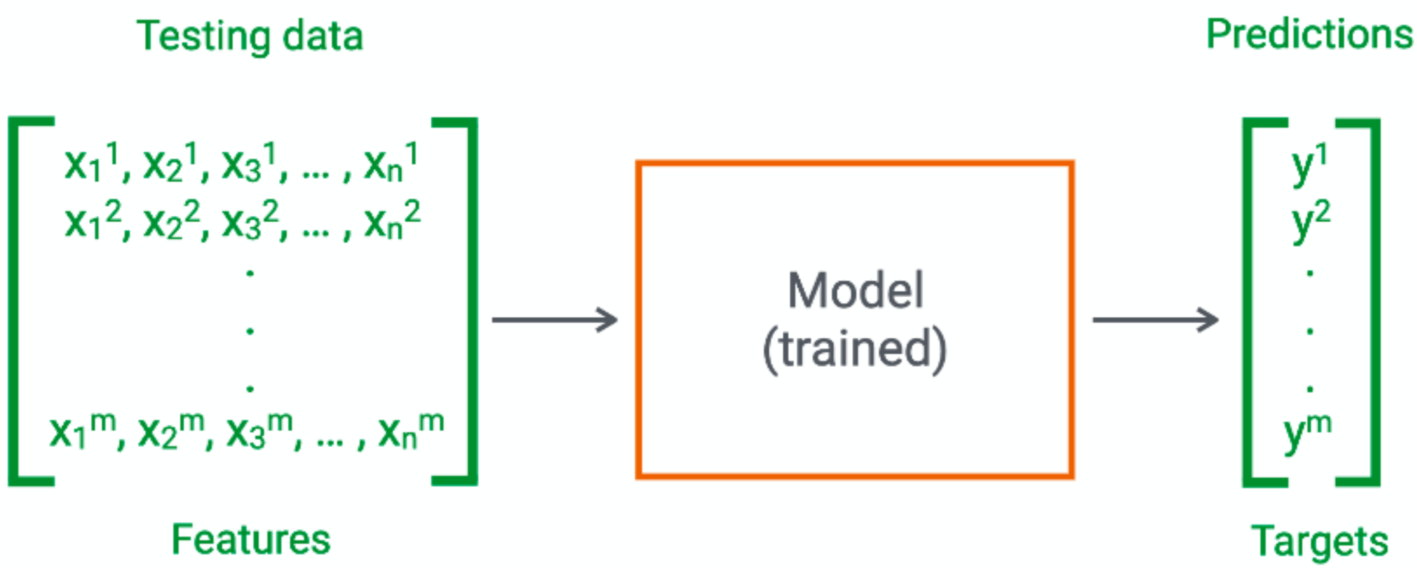
Training a model

First, you pick an model (algorithm) to use. You then pass the data to the model, and the model learns the parameters which best split your data



Making Predictions

Once the model is trained, you can use it to predict targets on unseen data

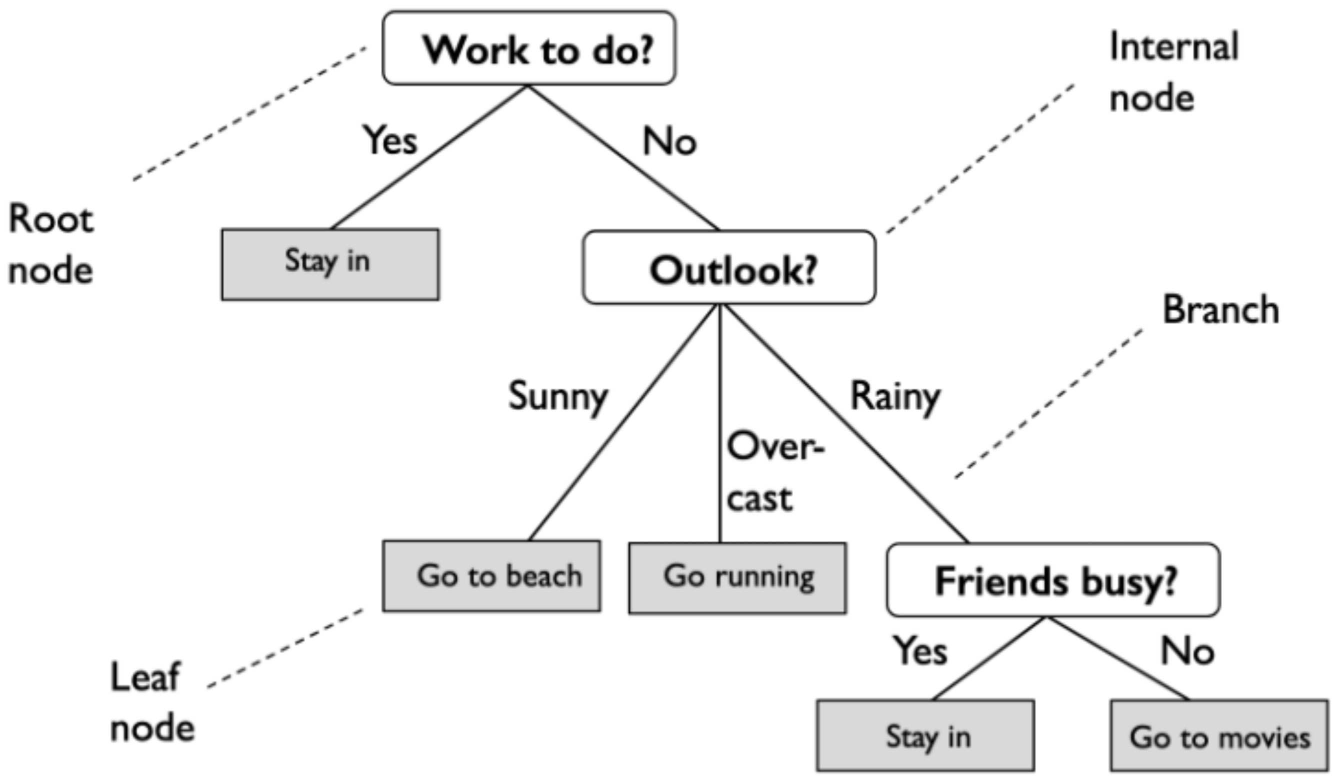


Algorithms

- More ML Algorithms Exist than I could talk about today
    - K-Nearest Neighbors: Search for clustering in data
    - Adaptive Linear Neurons: Building blocks for Deep Learning Algorithms
    - Support Vector Machines
    - Linear / Logistic Regression
    - Tree Based Learning
- Tree Based learning is the most popular in IceCube

Tree Based Learning

Tree Based Learning



Features of decision tree classifier

- Easy to understand and interpretable model
- Requires little data preparation
- Can model non-linear relationships
- Building block for more advanced models (e.g. random forests, boosted decision trees)

Constructing a decision tree: asking the right questions

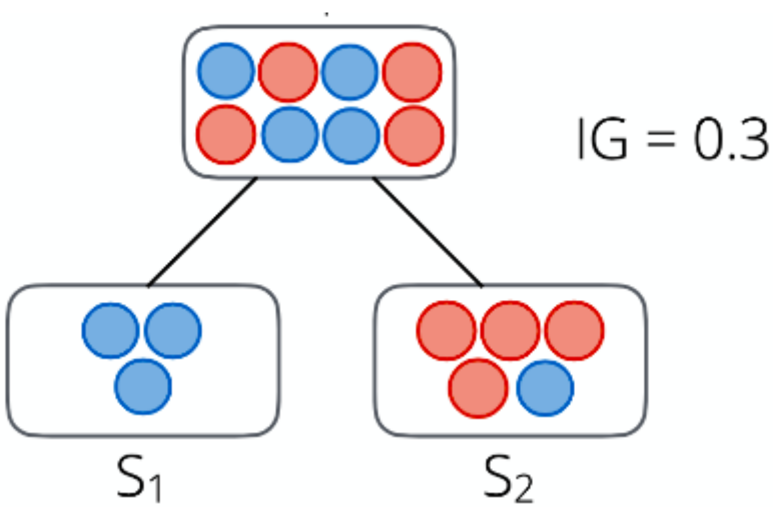
To construct a decision tree, you just have to choose a quantity to maximize (or minimize)

Example: Choose the splitting that maximizes the *information gain*, IG;

$$IG\left(S_p, f\right) = I\left(S_p\right) - \frac{N_1}{N_p} I\left(S_1\right) - \frac{N_2}{N_p} I\left(S_2\right)$$

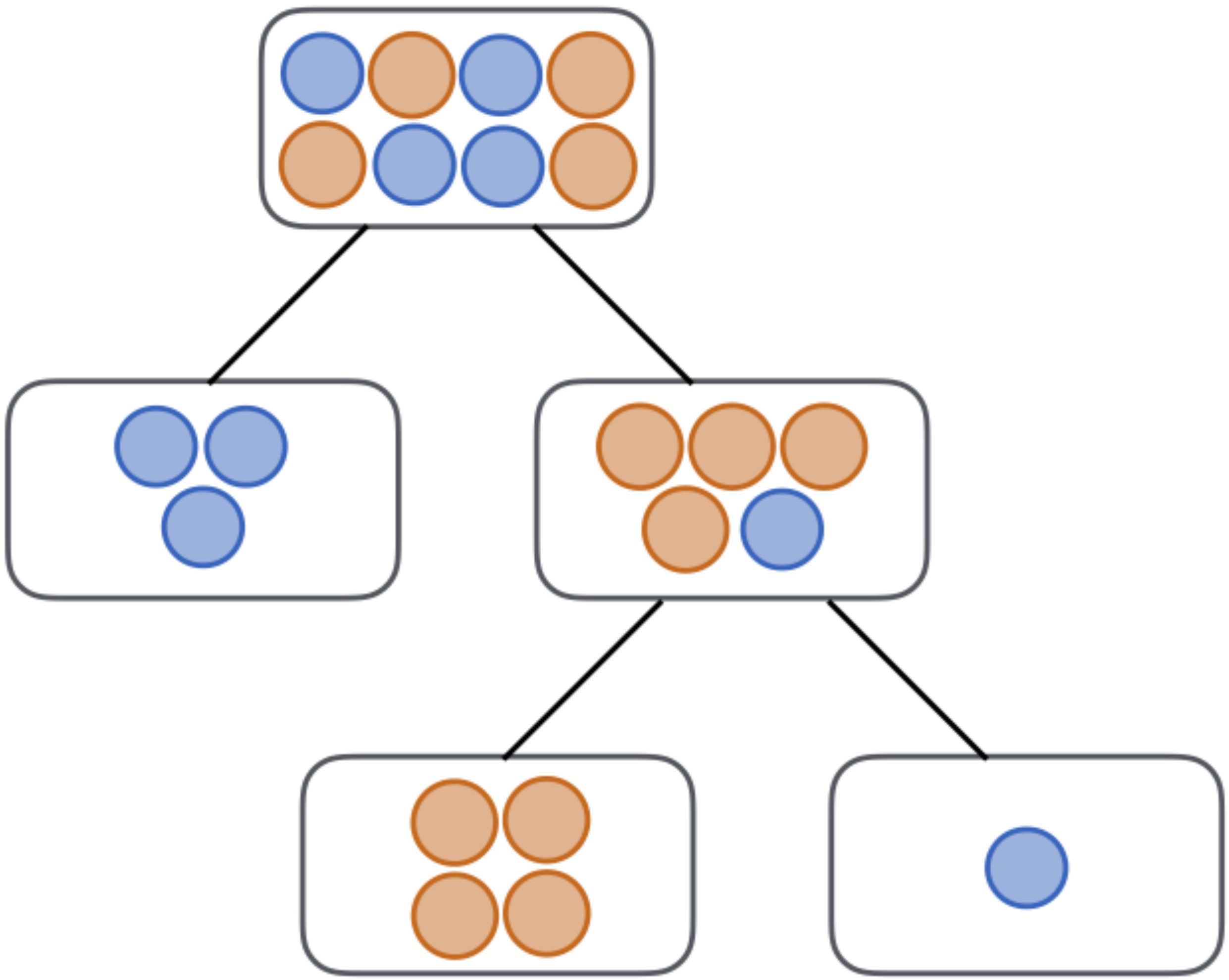
where the impurity, I, is defined as

$$I(S) = 1 - \sum_{i=1}^{N_{\text{class}}} p(i|S)^2$$



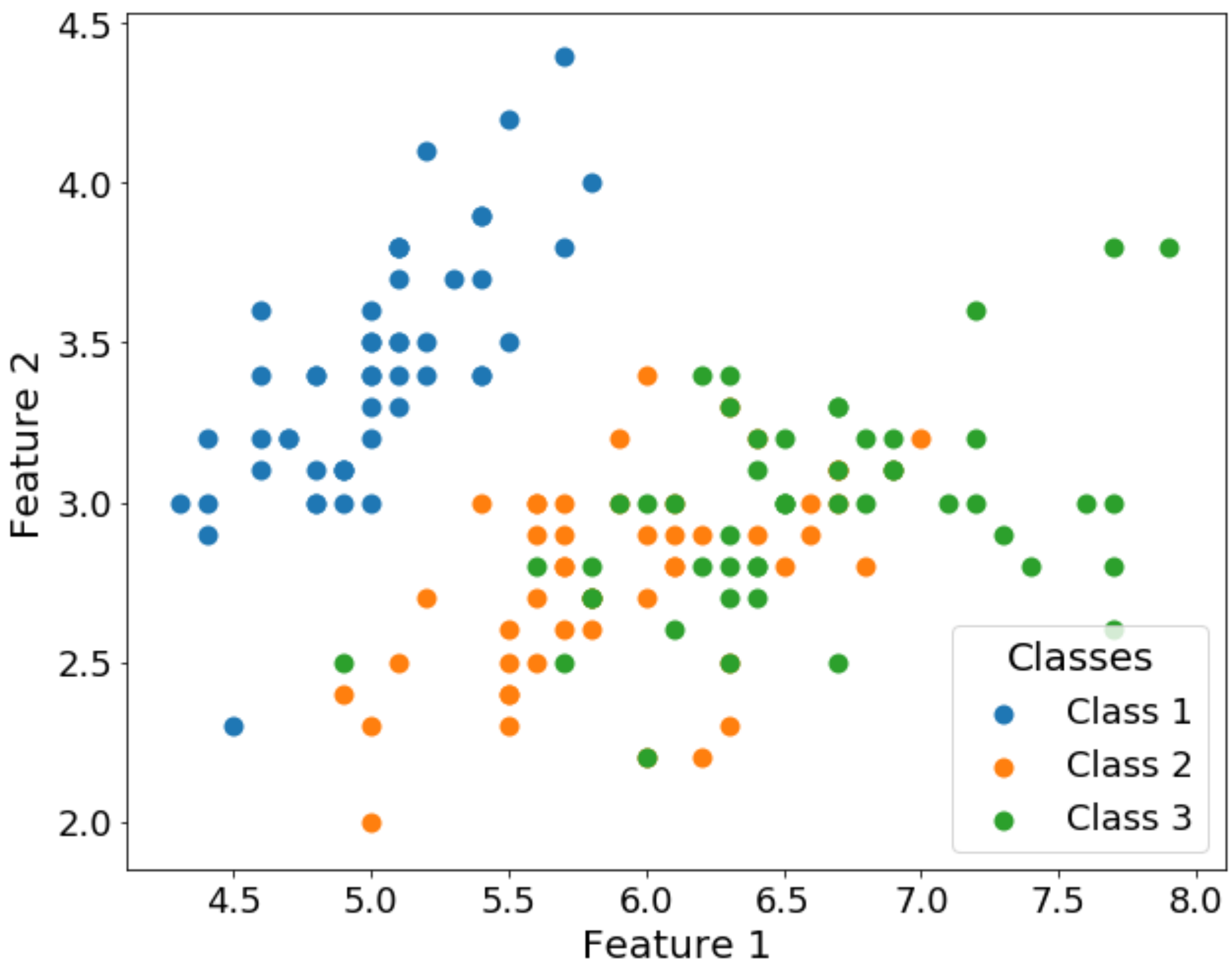
Constructing a decision tre

Continue doing this until each leaf is pure or until you've met other stop conditions



Visualizing decision trees

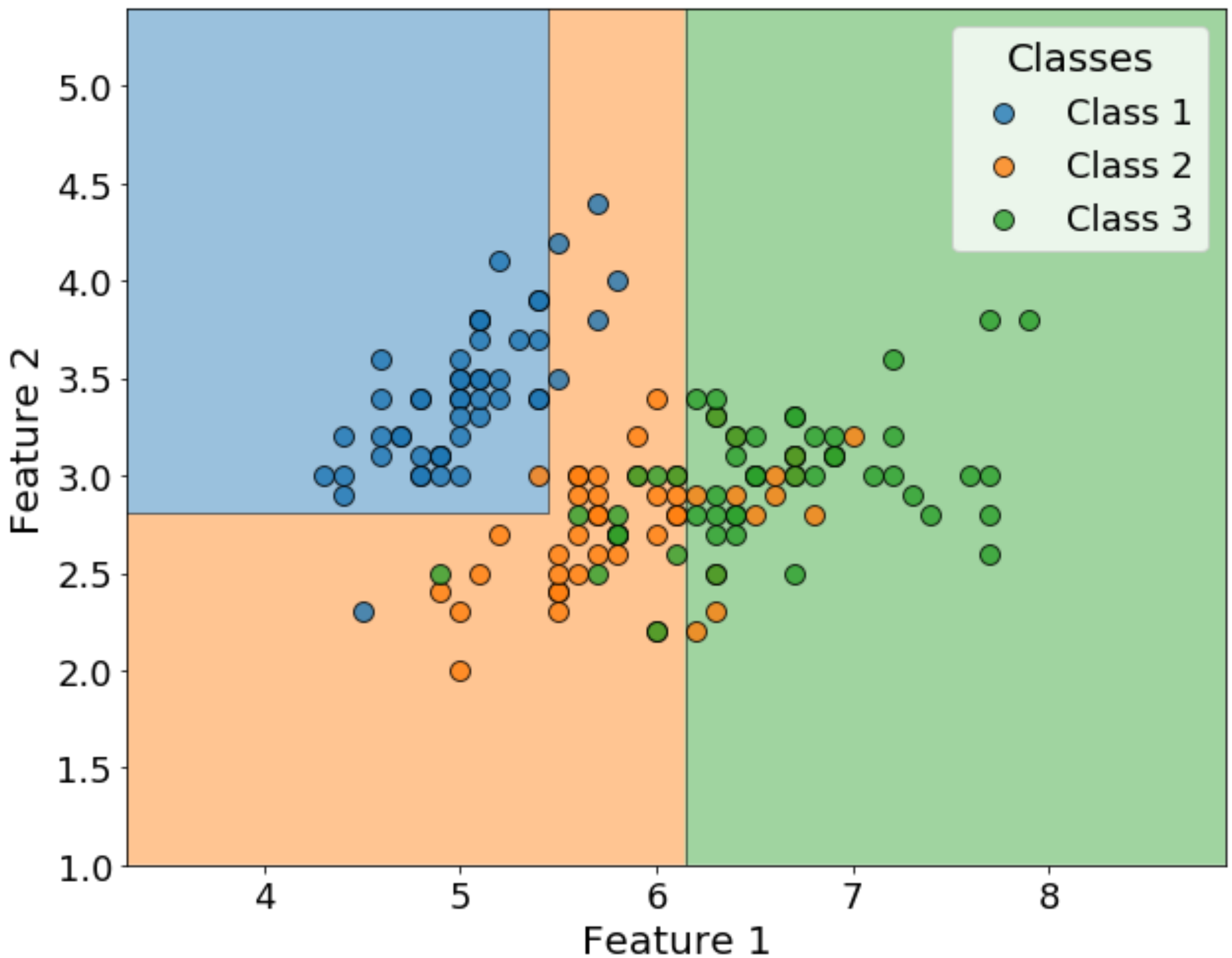
In [15]:
```
plotting.plot_2D_iris()
```



Visualizing decision trees

In [16]:

```
plotting.plot_tree_decision_regions(clf)
```



Ensemble Learning: Random Forests

Many times, individual algorithms (*weak learners*) are combined to create a more sophisticated algorithm

Idea: Combine many different decision trees to improve performance

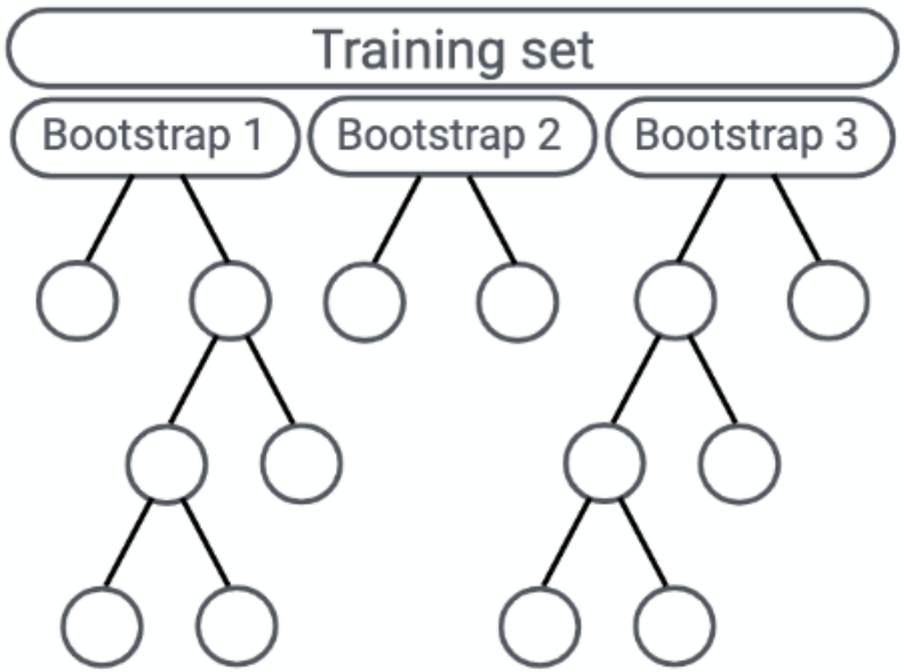
1. Split up your training data into *n bootstrap* samples
2. For each one of these samples, build a decision tree
    - Only use a subset of available features for each tree
3. Once all trees are built, classify testing data by majority vote from all your trees

Split up your training data into *n bootstrap* samples

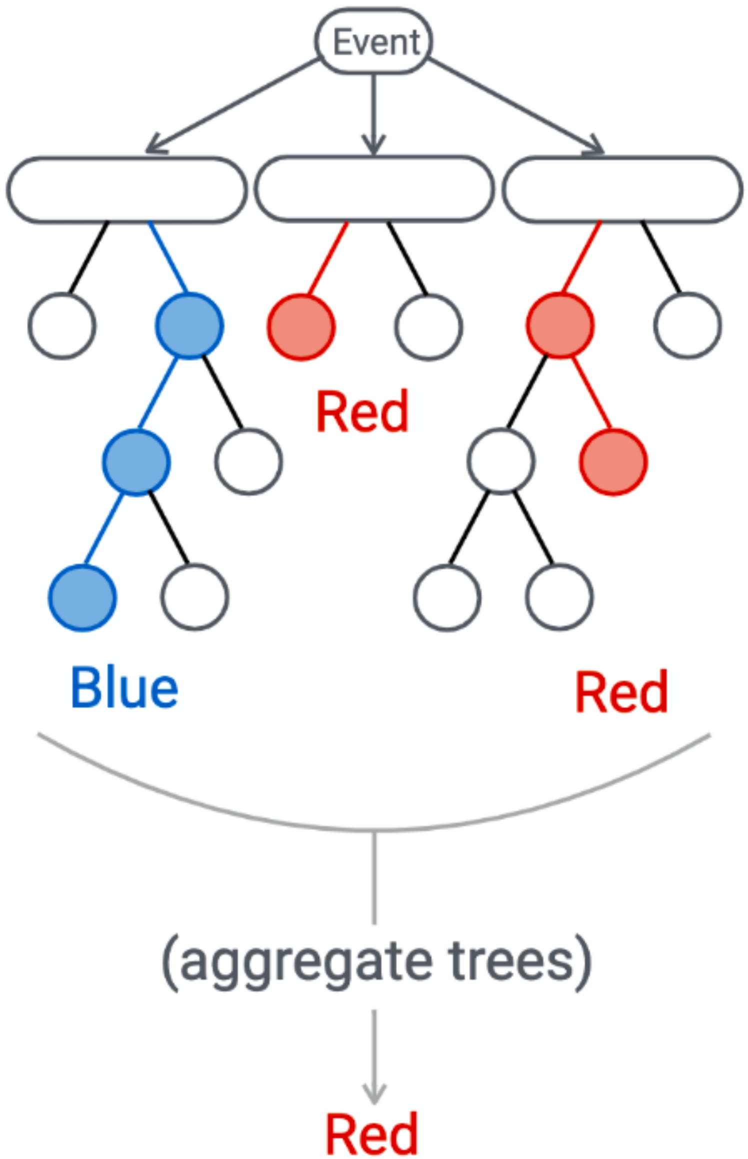
Split up your training data into *n bootstrap* samples

For each one of these samples, build a decision tree

```
- Only use a subset of available features for each tree
```



Once all trees are built, classify testing data by majority vote from all your trees

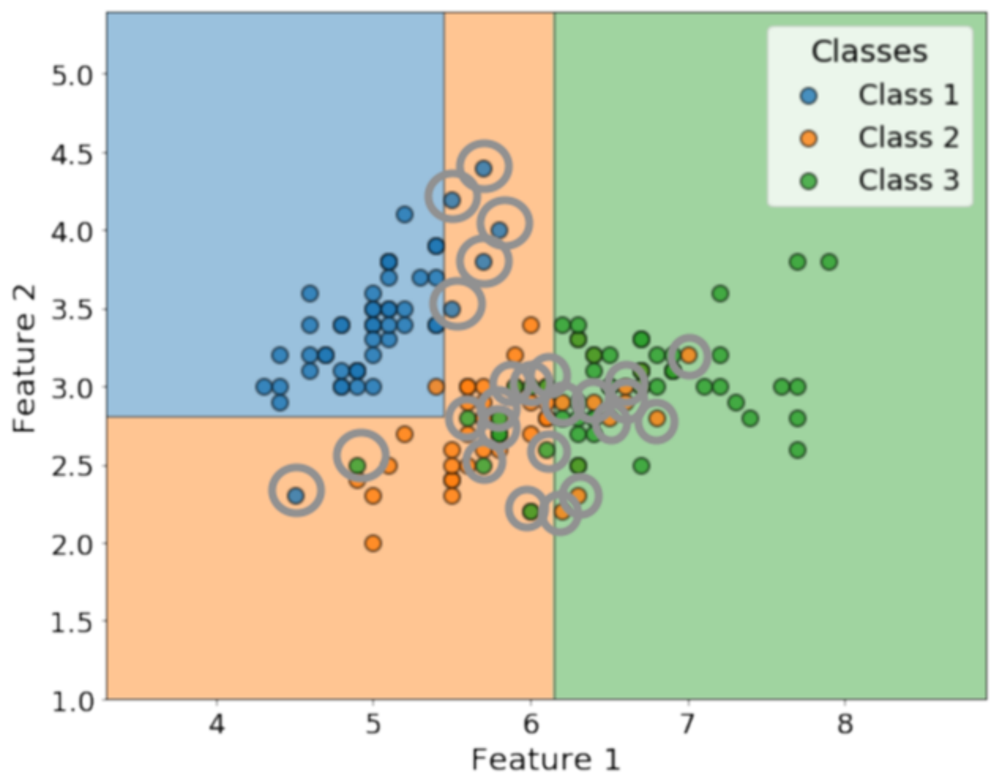


Ensemble Learning: Boosted Decision Trees

Boosted Decision Trees (BDTs) are similar to Random Forests, but they don't randomly draw "bootstrap" samples

Instead they find where the trees perform the *worst*, and *boost* the power of the next tree by weighting misclassifications more heavily

The result is a weighted average from many trees, instead of just a majority vote



Model Validation

Model Validation

You want a model that is complex enough to describe your data, but not too complex that it treats statistical fluctuations as meaningful.

- Under-fitting — model isn't sufficiently complex enough to properly model the dataset at hand
- Over-fitting — model is too complex and begins to learn the noise in the training dataset

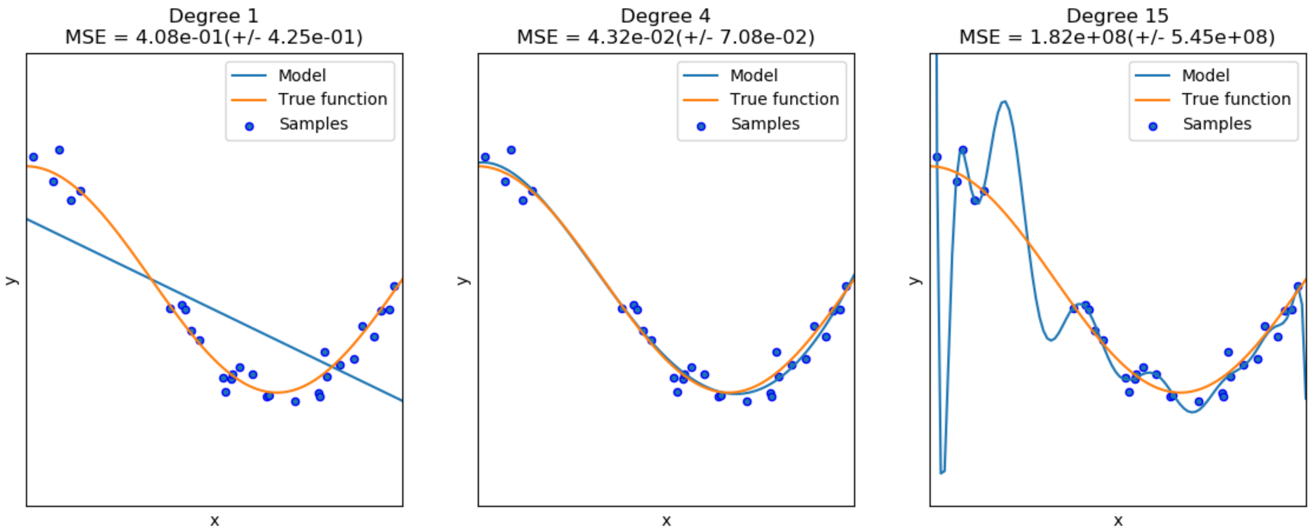


Image source: [Underfitting vs. Overfitting (http://scikit-learn.org/stable/auto_examples/model_selection/plot_underfitting_overfitting.html)](http://scikit-learn.org/stable/auto_examples/model_selection/plot_underfitting_overfitting.html) in scikit-learn examples

Model validation: hyperparameter tuning

Parameters unique to the algorithm you are using are called *hyperparameters*.

Changing hyperparameters can make a model more or less complex

For BDTs or random forests, these include:

1. The depth of a tree
2. The total number of trees
3. Number of features considered in each tree

Model validation: training & testing sets

- A trained model will generally perform better on data that was used to train it
- Want to measure how well a model generalizes to new, unseen data
- Need to have two separate datasets. One for training models and one for evaluating model performance

Model validation: *k*-fold cross validation

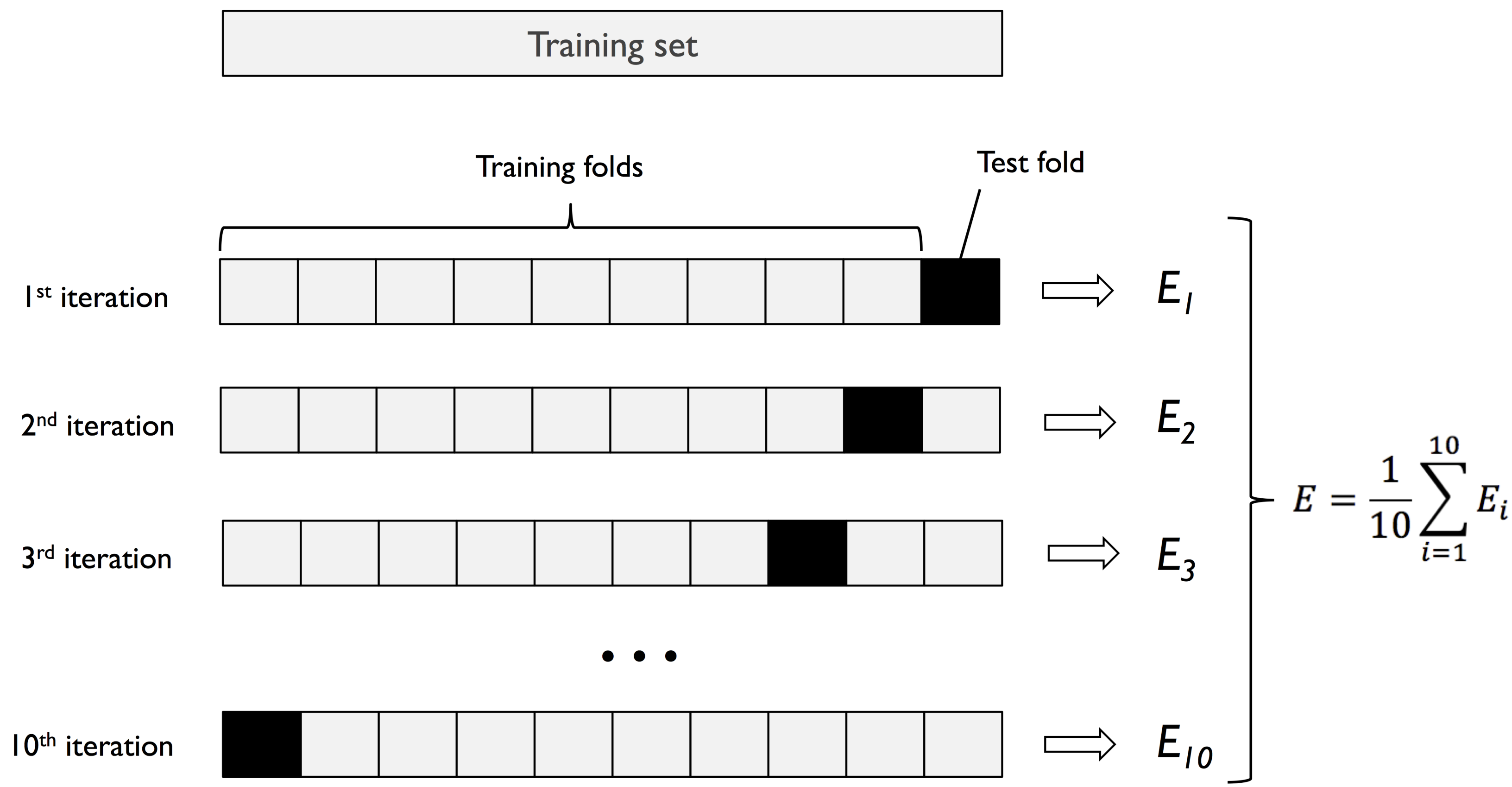
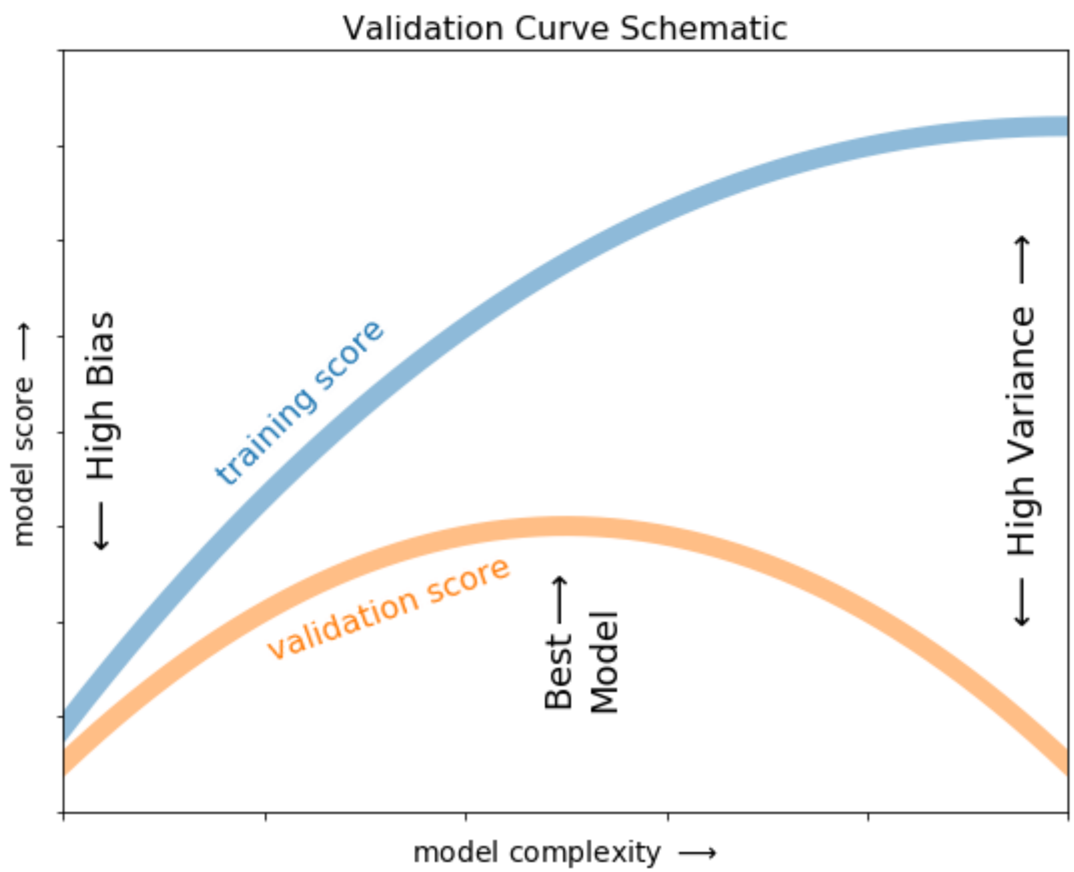


Image source: Raschka, Sebastian, and Vahid Mirjalili. [Python Machine Learning (https://www.amazon.com/Python-Machine-Learning-scikit-learn-TensorFlow/dp/1787125939)](https://www.amazon.com/Python-Machine-Learning-scikit-learn-TensorFlow/dp/1787125939), 2nd Ed. Packt Publishing, 2017.

Validation curves

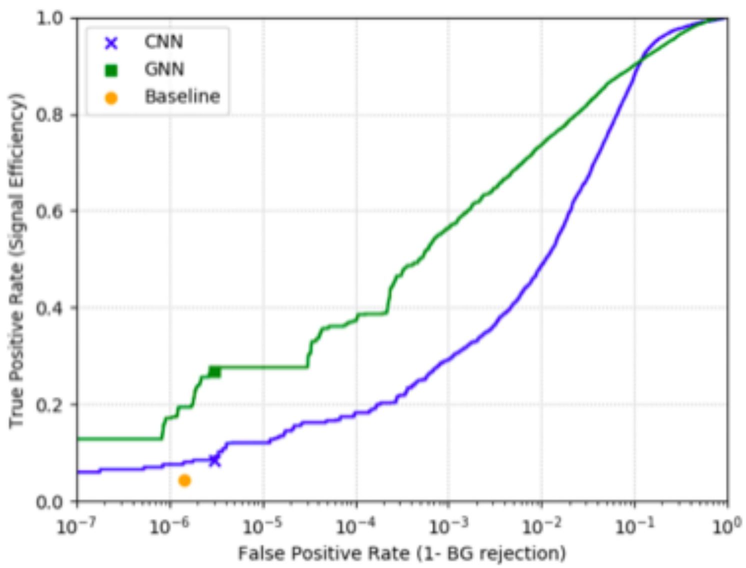
Validation curves are a good way to diagnose if a model is under- or over-fitting

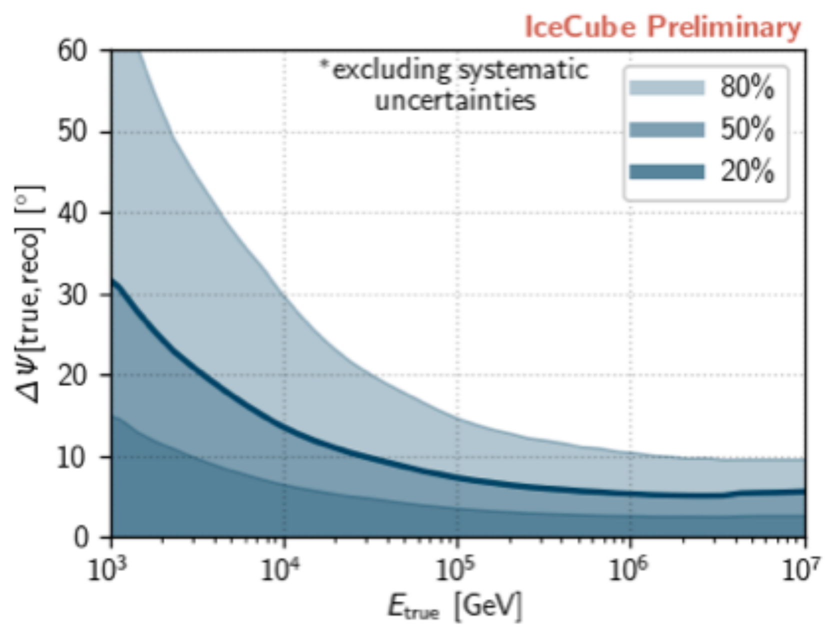
In [18]:
```
plotting.plot_validation_curve()
```



IceCube Use Cases

- BDTs: Event selections
- BDTs: Low-Energy Particle Identification
- Random Forests: Angular Error Estimation
- Deep Learning: Event Reconstructions





Additional Resources

- Notebook that works through decision tree and Random forest examples: GitHub (https://github.com/apizzuto/bootcamp-machine-learning/blob/master/decision_trees_and_nearest_neighbors.ipynb)
- *Python Machine Learning* by Sebastian Raschka: GitHub (https://github.com/rasbt/python-machine-learning-book-2nd-edition)
- *Data Science Handbook* by Jake VanderPlas: GitHub (https://github.com/jakevdp/PythonDataScienceHandbook)
- *The Elements of Statistical Learning* by Hastie, Tibshirani and Friedman: Free Book (https://web.stanford.edu/~hastie/ElemStatLearn/)
- *Deep Learning* by Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Amazon (https://www.amazon.com/Deep-Learning-Adaptive-Computation-Machine/dp/0262035618)

Thank you!

Questions?