##########################REVIEW DECISION TREES###########################

Decision Tree algorithms

1. try to understand the system
    a. split into subsets
    b. For each subset
        i. Is it pure? (all yes or no)
            1. if yes, stop
            2. if not split into subsets again
2. **Everyone try it!**
3. Start with outlook, create a decision tree
4. Follow decision tree: Rain High Weak
    a. What is his decision?

ID3 Algorithm builds the dataset

Split(node, {examples}):
1. Find best attribute to split on (A)
2. Decision attribute for this node (A)
3. For each value of A, create a new child node
4. Split training {examples} to child nodes
5. For each child node:
    if subset is pure: stop
    else: split(node, {examples})

How to know which attribute to split on?

Example:
    Look at original data, which attribute is best to split on?
    We like splits with pure subsets, or have a higher certainty
    4/0 -- completely certain (100%)
    3/3 -- completely uncertain (50%)

Use Entropy:
    $H(S) = -p_+ * \log_2(p_+) - p_- * \log_2(p_-)$
    S .. subset of training examples
    p_+/p_- .. % of positive / negative examples in S
    Interpretation: assume item X belongs to S
        -how many bits need to tell if X is positive or negative
        -lets try this with different values like .5,.5 and 1,0
        -log2(x) ... the power you need to raise 2 to get x

        GainSplit = Gain(s,a) = $H(S) - \sum(\frac{|Sv|}{|S|} * H(S_v))$

            V ... possible values of A
            S ... set of examples {X}
            S_v .. Subset where $X_a = V$

Explain with golf example

-Splitting on entropy, bias towards attributes with lots of values

-Large trees with many branches prefered

-Don't use ID as an attribute!

Splitting on Gain Ratio:

GainRatio = GainSplit/SplitInfo

SplitInfo = sum((n_i/n)log(n_i/n))

n_i = is the number of records in partition i

n = number of records before partition

Explain with golf example

How to avoid overfitting:

Stop before it becomes a fully grown tree

Normal Stopping conditions

stop if all instances belong to same class

stop if all the attribute values are the same

Early Stopping conditions

Stop if number of instances is less than a user specified threshold

Stop if class distribution of instances are independent of the available attributes

Stop if splitting the current node improves the impurity measure

(eg: information gain) below a given threshold

How to handle continuous attributes:

--Split them into groups and test it

Random Forest

--Grow K different trees

--pick a random subset and random attributes and use that to build a tree

--Given a new data point, classify it via all trees

use majority vote, class predicted most often

Pro's and Con's

Pros:

--Easy to classify unknown records

--Easy to interpret for small sized trees

--Able to handle both continuous and discrete attributes

--If you use a method to avoid over fitting it deals well with noise

--Deals well with outliers

--Shows which fields are most important

Con

--Irrelevant attributes may cause a bad tree

--Decisions are rectilinear
--Small variations in data can cause very different trees
--A subtree might be replicated many times
--Too many classes can cause errors
--Not good for predicting a continuous class attribute, needs to be categorical. You can create categories from a continuous attribute
--Algorithm greatly depends on how you do the split
--Overfitting:
    --trees more complex than necessary
    --Split into a test and training dataset
    --resubstitution errors, (error on training data)
    --generalization errors, (error on test data)

##
##

Show examples of Decision Tree and Gaussian Naive Bayes

Gaussian Naive Bayes

Naive Bayes is for categorical data
Solution to continuous data is Gaussian Naive Bayes
Common Approach: Assume P(Xi | y = yk) follows a gaussian distribution
So a Gaussian is used for the Probability Function:

$$p(X_i = x \mid Y = y_k) = \frac{1}{\sqrt{2\pi\sigma_{ik}^2}} exp(-\frac{1}{2} * (\frac{x-\mu_{ik}}{\sigma_{ik}})^2$$

If x is continuous and y is discrete
Find the mean and variance of the group of values in that category for mu and sigma^2

##
##
k-Nearest Neighbor

1. Find the k nearest neighbors of new point
2. assign point to class which it has the most neighbors for
3. if K = 1, it makes a voronoi partition around each item
    Voronoi Tesselation -partitions spaces into regions
    boundary: points at the same distance from two different training examples
    classification boundary
        non-linear, reflects classes well
        compare to NB, DT, Logistic
4. Thoughts:
    a. choose an odd value k for a 2 class problem
    b. k must not be a multiple of the number of classes

c. the main drawback is the complexity of searching for the nearest neighbor for each sample
d. can overfit really well, like if the data is mislabeled (solution is looking at multiple neighbors.

##
##

Show kNN Algorithm in sup1.py