CS 194-10, Fall 2011
Assignment 3

1. *Entropy and Information Gain*
   The entropy of a Bernoulli (Boolean 0/1) random variable X with $P(X = 1) = q$ is given by

   $$B(q) = -q \log q - (1 - q) \log(1 - q) \ .$$

   Suppose that a set S of examples contains p positive examples and n negative examples. The entropy of S is defined as $H(S) = B(\frac{p}{p+n})$

   (a) Show that $H(S) \leq 1$ and the equality is achieved when $p = n$.

   (b) Suppose that based on an attribute X_j we split our examples into disjoint subsets S_k, with p_k positive and n_k negative examples in each. The information gain of this attribute is given by

   $$\text{Gain}(S, X_j) = H(S) - \sum_k \frac{|S_k|}{|S|} H(S_k) \ .$$

   Show, using Lagrange multiplies or otherwise, that the attribute has strictly positive information gain unless the ratio $\frac{p_k}{p_k+n_k}$ is the same for all k.

2. *Empirical Loss and Splits*
   Can the empirical loss ever *increase* when splitting on an attribute? If so, give an example. If not, give a proof. Consider both 0/1 loss and L_2 loss.

3. *Splitting continuous attributes*
   In lecture an informal argument was given for the claim that an optimal split point for a continuous attribute always comes between examples with different Y-values, in the case where information gain is the criterion. Prove that the same claim holds for empirical loss.

4. *Majority voting*
   Consider a learning algorithm that uses a majority voting among K learned hypotheses. Suppose that each hypothesis has error ϵ and that the errors made by each hypothesis are independent of the others'.

   (a) Calculate a formula for the error of the ensemble algorithm in terms of K and ϵ.

   (b) If the independence assumption is removed, can the ensemble error be worse than ϵ ?

5. In this question you will apply decision tree learning to classify the phases of seismic detections using the same dataset as in A2, with the same attributes. You will learn one tree for each of the top 10 stations, the output of each being an 8-valued discrete variable with values Lg, P, PKP, Pg, Pn, Rg, S, Sn.

   (a) To build your decision trees, write Python code for the decision tree learning algorithm described in Russell & Norvig, Figure 18.5 (also in lecture slides), suitably modified to handle continuous attributes. You may use any attribute and split selection criterion you like, and any representation you like for the learned tree; but trees should be object of class `decision_tree`. After the tree is built, it must be pruned. To do this, use χ^2 pruning: working upwards from the bottom of the tree, prune each node (i.e., replace it by a suitable leaf node) if (1) all of its K children are leaves and (2) the *deviation* Δ of its children exceeds a predefined confidence bound c for the χ^2 statistic with $K - 1$ degrees of freedom. The deviation is defined as

   $$\Delta = \sum_{k=1}^{K} \sum_{\ell=1}^{L} \frac{(N_{k\ell} - \hat{N}_{k\ell})^2}{\hat{N}_{k\ell}}$$

for a discrete output variable with L values, where $N_{k\ell}$ is the number of examples in child k with output label ℓ and $\hat{N}_{k\ell}$ is the expected number under the null hypothesis that the attribute is irrelevant for predicting the output. Values for χ^2 are available from the `chisquare` function in `scipy.stats`. The desired confidence bound c affects the tradeoff between degree of fit and complexity of the learned tree; you can optimize it using cross-validation if you wish.

Once you have learned each decision tree, write it out to a file called `tree_`s, where s is the station ID, using `pickle.dump`. Report your estimate of its classification accuracy and the basis for this estimate. Since we don't know what format you are using for the tree, you will also need to supply a file `dtclassify.py` defining `classify(decision_tree,example)` which returns the predicted output value (here, a phase label) for the example according to the tree.

(b) Now apply the *bagging* algorithm (Hastie *et al.*, Section 8.7) to improve classification accuracy. For each of the 10 stations, train M independent decision tree classifiers from bootstrap samples of size N drawn with replacement from the original data; the final prediction for a new example is made by majority voting across the M outputs. As always, the value of M can be optimized. As before, write the collected classifiers out to a file as a single object of type `bagged_decision_tree` (could be as simple as a list of trees) and supply a function `bagged_classify(bagged_decision_tree,example)` in your `dtclassify.py` file. Report your estimate of its classification accuracy and the basis for this estimate.

Submit your files collected together as `a3.firstname.lastname.tar.gz` (for one person) or `a3.firstname1.lastname1.`
(for two people).