Homework assignment 3:
Decision trees and ROC curves

Objective: The objective of this exercise is to become familiar with fitting decision trees and making ROC curves in Python.

Material: Pang-Ning Tan, Michael Steinbach, and Vipin Kumar, *Introduction to Data Mining*, section 4.1-4.6, as well as the included article on ROC curves.

Important: When handing in your homework:

- Provide clear and complete answers to the questions below (not hidden somewhere in your source code), and make sure to explain your answers / motivate your choices. Please make as PDF file.
- Source code, output graphs, derivations, etc., should be included, and zipped together with the PDF file.
- Hand-in: upload to Blackboard.
- Include name, student number, assignment (especially in filenames)
- When working in pairs both of you should upload the assignment, and report the name of your partner in the upload comments.
- For problems or questions: use the BB discussion board or email.

3.1 Decision trees

In this part of the exercise we will fit decision trees using the scikitlearn classifier `sklearn.tree.DecisionTreeClassifier`. As a splitting criterion, the function uses one of the following two impurity measures:

$$\begin{aligned} \mathrm{gdi}(t) &= 1 - \textstyle\sum_{i=1}^{C} p(i|t)^2 && \text{equivalent to } \mathrm{Gini}(t) \\ \mathrm{deviance}(t) &= -2 \textstyle\sum_{i=1}^{C} p(i|t) \log p(i|t) && \text{equivalent to } \mathrm{Entropy}(t) \end{aligned}$$

We will analyze the wine data we have used previously.

The wine data set has the following attributes, all of which are continuous:

This exercise is based upon material kindly provided by the Cognitive System Section, DTU Compute, `http://cogsys.compute.dtu.dk`. Any sale or commercial distribution is strictly forbidden.

Wine data set

#	Attribute	Unit
1	Fixed acidity (tartaric)	g/dm^3
2	Volatile acidity (acetic)	g/dm^3
3	Citric acid	g/dm^3
4	Residual sugar	g/dm^3
5	Chlorides	g/dm^3
6	Free sulfur dioxide	mg/dm^3
7	Total sulfur dioxide	mg/dm^3
8	Density	g/cm^3
9	pH	pH
10	Sulphates	g/dm^3
11	Alcohol	% vol.

3.1.1 Load the wine data set `Data/wine.mat` into Python. This contains the same data as used in the earlier assignment, but with outliers and the 12th attribute already removed.

Hints:

- The attributes are stored in matrix `X`, the class in vector `y`. Attribute names and class names are stored in the attributeNames and classNames variables.

3.1.2 Fit a decision tree to the wine data in order to estimate if the wine is red or white. Use the Gini (gdi) splitting criterion. Use `min_samples_split=100` for the stopping criterion. Explain what happens when you change the values of the parameter `min_samples_split`. After fitting the tree, export its structure to GraphViz format to visualize it.

Hints:

- You will need to install the scikit-learn package if you haven't already. This can be found at `http://scikit-learn.org/stable/`.
- Help documentation for the DecisionTreeClassifier function can be found at `http://scikit-learn.org/stable/modules/tree.html#classification`
- The GraphViz editor for visualising the tree can be found at `http://www.graphviz.org/`

3.1.3 Show that a wine with the following attribute values would be classified as white by the tree fitted in 3.1.2.

#	Attribute	Value
1	Fixed acidity (tartaric)	6.9 g/dm^3
2	Volatile acidity (acetic)	1.09 g/dm^3
3	Citric acid	0.06 g/dm^3
4	Residual sugar	2.1 g/dm^3
5	Chlorides	0.0061 g/dm^3
6	Free sulfur dioxide	12 mg/dm^3
7	Total sulfur dioxide	31 mg/dm^3
8	Density	0.99 g/cm^3
9	pH	3.5
10	Sulphates	0.64 g/dm^3
11	Alcohol	12 %

Which of the 11 attributes are actually used in classifying this particular wine?

Hints:

- If you don't know how to classify input values with a tree, see the help documentation for the DecisionTreeClassifier function.

3.1.4 Classify all the wines in the wine data set. What percentage of the wine data is classified correctly by the tree?

3.2 Decision tree pruning using cross-validation

In this exercise we will use cross-validation to prune a decision tree. When applying cross-validation the observed data is split into training and test sets, i.e., `X_train`, `y_train` and `X_test` and `y_test`. We train the model on the training data and evaluate the performance of the trained model on the test data.

3.2.1 Load the wine data set `Data/wine.mat` using the `loadmat()` function. Divide the data into a training and a test data set. Fit a decision tree to the training data using the Gini (`gdi`) splitting criterion.

Now, we want to find optimally pruned decision tree, by modifying its maximum depth. For different values of parameter `depth` (from 2 to 20) fit the decision tree, and compute the classification error on the training and test set (holdout cross-validation). Plot the training and test classification error as a function of the tree depth.

Hints:

- Take a look at the module `sklearn.cross_validation` and see how it can be used to partition the data into a training and a test set (holdout validation, `train_test_split()` function). Note that the package also

contains functions to partition data for K-fold cross-validation. Some of the functions can ensure that both training and test sets have roughly the same class proportions.

What appears to be the optimal tree depth? Do you get the same result when you run your code again, generating a new random split between training and test data?

3.2.2 Repeat the exercise above, using 10-fold cross-validation. To do this, you must divide the data set into 10 random training and test folds (make sure the splits are indeed random). For each fold, fit a decision tree on the training set and evaluate its performance on the test set. Finally, compute the average classification error across the 10 cross-validation folds and plot it as a function of the tree depth.

Hints:

- This time the `KFold()` function from the module `sklearn.cross_validation` can be used to partition the data into the 10 training and test partitions.

What appears to be the optimal tree depth? Do you get the same result when you run your code again, generating a new random split between training and test data? How about 100-fold cross-validation?

3.3 ROC curves, AUC scores, and the sign test

In this exercise we will use ROC curves and the sign test to compare classifiers. Study the lecture slides and the paper 'ROC Graphs: Notes and Practical Considerations for Researchers' by Tom Fawcett included with the homework assignment (ROC101.pdf). It describes all you need to know (and more) about ROC curves. The method explained for computing the area under the curve is unnecessarily complicated. A simpler formula is:

$$\text{AUC} = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} \mathbf{1}_{p_i > p_j} .$$

Here i runs over all m data points with true label 1, and j runs over all n data points with true label 0; p_i and p_j denote the probability score assigned by the classifier to data point i and j, respectively. $\mathbf{1}$ is the indicator function: it outputs 1 if the condition (here $p_i > p_j$) is satisfied and 0 otherwise.

3.3.1 Load the file `Data/classprobs.xls`. The first column gives the true class label (either 0 or 1). The second and third column give the probabilistic scores for two different classifiers. The higher this probability, the more certain the classifier is that the example belongs to class 1 (instead of class 0).

3.3.2 Calculate the ROC curves for the classifiers and plot them. Interpret the obtained results. Do both classifiers perform better than random guessing?

Hints:

- The function `sklearn.metrics.roc_curve` can be used for computing the ROC.

3.3.3 Compute the AUC scores (area under the curve) of both classifiers using the formula given above. Do the AUC scores indicate that the classifiers are performing better than this baseline?

3.3.4 Using a threshold of 0.5, translate the probability scores to predicted class labels, and compute the accuracy for each of the classifiers.

3.3.5 Create the 2x2 table of wrong and correct classifications by the first and the second classifier, like in the lecture slides. Perform a sign test to test whether the performance of the two classifiers is significantly different at a significance level of 0.05.

Hints:

- As explained in the lecture slides, the sign test is a binomial test on the lower-left ($N_{1<2}$) and upper-right ($N_{1>2}$) elements of the cross table. Unlike in the lecture slides, here you need to perform a two-sided test: $\text{p-value} = P(W \leq \min(N_{1<2}, N_{1>2}) \text{ or } W \geq \max(N_{1<2}, N_{1>2})$.
- The function `scipy.stats.binom.cdf` can be used to compute the cumulative density of the binomial distribution.
- For more information on the sign test, see the included paper by Salzberg (signtest.pdf, in particular section 3.1).