LAB Artificial Intelligence

Programming Assignment 5 – Decision Trees and Random Forests

You are required to submit your solutions of this assignment on Moodle, before the next lab time. Submit your own work. Cheating will not be tolerated and will be penalized.

Decision Trees and Random Forests

I. Decision Trees

Decision Tree is a supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

Decision trees are simple to understand and to interpret. Trees can be visualized. But a decision tree can overfit easily and fail to generalize well to new samples.

II. Random Forests

Random Forest is one of the most common ensemble methods, which consists of a collection of Decision Trees.

The idea behind a Random Forest is very simple: we repeatedly select data from the data set (with replacement) and build a Decision Tree with each new sample. It is important to note that since we are sampling with replacement, many data points will be repeated and many will not be included as well. Since each tree does not get a chance to see all of the training data, so the unseen data can be used to cross validate each tree individually.

Each tree is grown as follows:

1. If the number of cases in the training set is `N`, sample `N` cases at random - but *with replacement*, from the original data. This sample will be the training set for growing the tree.
2. If there are `M` input variables, a number `m<<M` is specified such that at each node, `m` variables are selected at random out of the `M` and the best split on these `m` is used to split the node. The value of `m` is held constant during the forest growing.
3. Each tree is grown to the largest extent possible. There is no pruning.

Thus, the adjustable parameters when building a forest are:

- `N`: the number of trees
- `m`: the number of features per tree. A good value for `m` is the square root of the total number of features.

To classify a new object from an input vector, put the input vector down each of the trees in the forest. Each tree gives a classification, and we say that the tree "votes" for that class. The forest chooses the classification having the most votes (over all the trees in the forest).

One of the main advantages of Random Forests model is that it does not overfit.

sklearn: scikit-learn library

For this lab, we are going to use **sklearn,** which is scikit-learn library for SciPy. It is an open source machine learning library that supports supervised and unsupervised learning. It also provides various tools for model fitting, data preprocessing, model selection and evaluation, and many other utilities.

Scikit-learn assumes a very basic working knowledge of machine learning practices (model fitting, predicting, cross-validation, etc.). It provides dozens of built-in machine learning algorithms and models, called estimators. Each estimator can be fitted to some data using its fit method.

The fit method generally accepts 2 inputs:

- The samples matrix `X`. The size of `X` is typically (n_samples, n_features), which means that samples are represented as rows and features are represented as columns.
- The target values `y` which are real numbers for regression tasks, or integers for classification (or any other discrete set of values).

Both X and y are usually expected to be numpy arrays or equivalent array-like data types, though some estimators work with other formats such as sparse matrices.

Once the estimator is fitted, it can be used for predicting target values of new data. You do not need to re-train the estimator

1- 'Kyphosis' problem using sklearn

In this part, you are going to train a model that can predict whether a child has Kyphosis after their spinal surgery, using **sklearn**. The Kyphosis dataset has 3 features: `Age` of the child (in months), `Number` (of vertebrae) and the `Start` (top vertebra) operated on. You are going to train two models using the same dataset: decision tree classifier and random forest classifier.

Start by opening the file DT_Kyphosis.ipynb. This file contains a fully working example of the Kyphosis problem using sklearn. It contains all the code along with all the explanation needed. The first part of this lab is to familiarize yourself with the sklearn library functions.

Take your time to read every line in the code and understand it. Once done, move on to the second part of the lab where you are going to use sklearn to solve the Loans prediction problem.

NB: The DT_Kyphosis.ipynb contains a visualization part for the decision tree. The simplest way to visualize the tree is by using the `plot_tree` built-in function in sklearn. A more sophisticated visualization can be done by using the `pydot` library. This second visulaization's code is given to you and commented. You can install this `pydot` library and run this visualization at home.

2- 'Loans' problem using sklearn

In this second part, you are going to train a decision tree classifier and a random forests classifier, in order to predict if a customer is going to fully pay their loans back or not.

Open the DT_Loans.ipynb file. Inspired by the Kyphosis problem, complete the missing parts of the code in this file. All the steps needed to solve this problem are explained to you in detail in the file.

Questions

Answer the following questions for the '**Loans' problem**:

1. Why is it that every time your rerun your complete code, you get different results?

2. We saw in the Loans problem that we need to replace the 'purpose' column by numerical values. Explain why we did not have to do the same thing in the 'Kyphosis' problem for the 'Kyphosis' column.

3. Complete the following table, as indicated by the following steps:
   - Start by running your complete code using `n_estimators=1` for the `RandomForestClassifier`. Write down the obtained values for DT and RF in the first two rows of the following table.
   - Rerun several times the Random Forest part ONLY, every time increasing the value of `n_estimators` (as indicated by the following table) and write down the obtained results.
   - Using the results of this table, what performed better the random forest or the decision tree? Explain in detail (give at least 2 interpretations).

			Weighted average		
		accuracy	**Precision**	**Recall**	**F1-score**
	DT				
n_estimators	**1**				
	10				
	100				
	1000				
	10000				

4. From the results you obtained in part 3, you may see that the predictions can/might be somehow improved. 'Feature engineering' can be a good way to do it. Conduct some research about the "feature engineering" process and explain how it could improve your model.