***Student: Victor Pugliese***

***R#: 11492336***

***Course: Independent Study on Python Machine Learning for Petroleum Engineering Application (PETR 5000)***

***Self-Homework #6***

1. ***What is the approximate depth of a Decision Tree trained (without restrictions) on a training set with 1 million instances?***

Decision trees are generally approximately balanced, so traversing the Decision Tree requires going through roughly

1. ***Is a node’s Gini impurity generally lower or greater than its parent’s? Is it generally lower/greater, or always lower/greater?***

The node’s Gini impurity is generally lower than its parent.

1. ***If a Decision Tree is overfitting the training set, is it a good idea to try decreasing* max\_depth*?***

If the Decision Tree is overfitting the training set, it is a good idea to try decreasing max\_depth.

1. ***If a Decision Tree is underfitting the training set, is it a good idea to try scaling the input features?***

Decision Trees are not affected by scaled features, doing that is not necessary.

1. ***If it takes one hour to train a Decision Tree on a training set containing 1 million instances, roughly how much time will it take to train another Decision Tree on a training set containing 10 million instances?***
2. ***If your training set contains 100,000 instances, will setting* presort=True *speed up training?***

Presorting the training set speeds up the algorithm is the dataset is smaller than a few thousand instances, conversely it will slow down the process.

1. ***Train and fine-tune a Decision Tree for the moons dataset.***

***a. Generate a moons dataset using make\_moons(*n\_samples=10000, noise=0.4*).***

***b. Split it into a training set and a test set using* train\_test\_split().**

***c. Use grid search with cross-validation (with the help of the* GridSearchCV *class) to find good hyperparameter values for a DecisionTreeClassifier. Hint: try various values for max\_leaf\_nodes.***

***d. Train it on the full training set using these hyperparameters and measure your model’s performance on the test set. You should get roughly 85% to 87% accuracy.***

See file: HML\_Chapter06\_Exercise\_07.py

1. ***Grow a forest.***

***a. Continuing the previous exercise, generate 1,000 subsets of the training set, each containing 100 instances selected randomly. Hint: you can use Scikit-Learn’s* ShuffleSplit *class for this.***

***b. Train one Decision Tree on each subset, using the best hyperparameter values found above. Evaluate these 1,000 Decision Trees on the test set. Since they were trained on smaller sets, these Decision Trees will likely perform worse than the first Decision Tree, achieving only about 80% accuracy.***

***c. Now comes the magic. For each test set instance, generate the predictions of the 1,000 Decision Trees, and keep only the most frequent prediction (you can use SciPy’s* mode() *function for this). This gives you* majority-vote predictions *over the test set.***

***d. Evaluate these predictions on the test set: you should obtain a slightly higher accuracy than your first model (about 0.5 to 1.5% higher). Congratulations, you have trained a Random Forest classifier!***

See file: HML\_Chapter06\_Exercise\_08.py