C S 487/519 Applied Machine Learning I
Compare classifiers in scikit-learn library

1 Objective

In this *individual* project assignment, you are required to understand and compare several classification algorithms that are provided by the Python scikit-learn library.

2 Requirements

- (50 points) Write classification code by utilizing several scikit-learn classifiers: (i) perceptron, (ii) support vector machine (linear, and non-linear using Radial Basis Function (RBF) kernel), (iii) decision tree, and (iv) *K*-nearest neighbor. In total there are four classifiers.
- (15 points) Each classifier needs to be tested using two datasets: (1) the `digits` dataset offered by scikit-learn library, and (2) one dataset containing time-series instances. Example of the second dataset can be the REALDISP Activity Recognition Dataset.
- (15 points) Properly analyze the classifiers' behavior by applying the knowledge that we discussed in class. Such analysis should include at least accuracy and running time.
- (15 points) Understand the source code of `DecisionTreeClassifier` (You can follow the `source` link).
  - (5 points) Please denote **two** strategies that this classifier implements to pre-prune or post-prune the tree.
  - (10 points) For each strategy, please clearly identify the repository file and the lines of code that implement such strategies.
  - Put your understanding in a report file (**report.pdf**). The file content should be succinct.
- (5 points) Write a readme file **readme.txt** with the commands to run your code. Your code needs to run in command line, accepting as input parameters the classifier name, the dataset filename, and any required parameter. For example, "`python main.py knn dataset.csv -k 3`"
- Your Python code should be written for Python version 3.5.2 or higher.
- Please properly organize your Python code (e.g., create proper classes, modules). Each required task had better be implemented in a separate python file and imported into the main script. For example, to use the support vector machine classifier, you can create the script **mysvm.py** for both the linear and the nonlinear SVM implementation, then use "`import mysvm`" in the main.py file to test your implementation.

3 Submission instructions

Compress your python code and readme file to a zip file named **hw.zip** and upload it to Canvas.

4 Grading criteria

(1) The score allocation has already been put beside the questions.
(2) Please make sure that you test your code thoroughly by considering all possible test cases. Your code may be tested using more datasets.
(3) 5 points will be deducted if submitted files (including files types, file names, etc.) do not follow the instructions.
(4) If the total points are more than 100. Your grades will be scaled to the range of [0,100].