Review the assignment due date

Assignment 1

Total 22 marks -> would be scaled to 11 marks

Questions

1. Complete the decision tree implementation in tree/base.py. **[4 marks]** The code should be written in Python and not use existing libraries other than the ones shared in class or already imported in the code. Your decision tree should work for four cases: i) discrete features, discrete output; ii) discrete features, real output; iii) real features, discrete output; real features, real output. Your decision tree should be able to use GiniIndex or InformationGain (Entropy) as the criteria for splitting for discrete output. Your decision tree should be able to use InformationGain (MSE) as the criteria for splitting for real output. Your code should also be able to plot/display the decision tree.

   > *You should be editing the following files.*

   - `metrics.py` : Complete the performance metrics functions in this file.
   - `usage.py` : Run this file to check your solutions.
   - tree (Directory): Module for decision tree.
     - `base.py` : Complete Decision Tree Class.
     - `utils.py` : Complete all utility functions.
     - `__init__.py` : **Do not edit this**

   > *You should run usage.py to check your solutions.*

2. Generate your dataset using the following lines of code

```
from sklearn.datasets import make_classification
X, y = make_classification(
n_features=2, n_redundant=0, n_informative=2, random_state=1,
n_clusters_per_class=2, class_sep=0.5)

# For plotting
import matplotlib.pyplot as plt
plt.scatter(X[:, 0], X[:, 1], c=y)
```

   a) Show the usage of *your decision tree* on the above dataset. The first 70% of the data should be used for training purposes and the remaining 30% for test purposes. Show the accuracy, per-class precision and recall of the decision tree you implemented on the test dataset. **[1 mark]**

   b) Use 5 fold cross-validation on the dataset. Using nested cross-validation find the optimum depth of the tree. **[1 mark]**

   > *You should be editing* `classification-exp.py` *for the code containing the above experiments.*

3. a) Show the usage of your decision tree for the automotive efficiency problem. **[0.5 marks]**

   b) Compare the performance of your model with the decision tree module from scikit learn. **[0.5 marks]**

   > *You should be editing `auto-efficiency.py` for the code containing the above experiments.*

4. Create some fake data to do some experiments on the runtime complexity of your decision tree algorithm. Create a dataset with N samples and M binary features. Vary M and N to plot the time taken for: 1) learning the tree, 2) predicting for test data. How do these results compare with theoretical time complexity for decision tree creation and prediction. You should do the comparison for all the four cases of decision trees. **[2 marks]**

   > *You should be editing `experiments.py` for the code containing the above experiments.*

You must answer the subjectve questions (timing analysis, displaying plots) by creating `assignment_q<question-number>_subjective_answers.md`

Human Activity Recognition (Mini Project)

Human Activity Recognition (HAR) refers to the capability of machines to identify various activities performed by the users. The knowledge acquired from these recognition systems is integrated into many applications where the associated device uses it to identify actions or gestures and performs predefined tasks in response.

Dataset

For this assignent we will be using a publically available dataset called UCI-HAR. The dataset is available to download here. The Dataset contains data for 30 participants . Each participant performed six activities while wearing a Samsung Galaxy S II smartphone on their waist (The video of the participants taking data is also available here). The smartphone's accelerometer and gyroscope captured 3-axial linear acceleration and 3-axial angular velocity. Read all the `readme` and `info` files for more information.

Preprocessing.

We will use the raw accelerometer data within the inertial_signals folder. The provided script, `CombineScript.py`, organizes and sorts accelerometer data, establishing separate classes for each category and compiling participant data into these classes. `MakeDataset.py` script is used to read through all the participant data and create a single dataset. The dataset is then split into train,test and validation set. We focus on the first 10 seconds of activity, translating to the initial 500 data samples due to a sampling rate of 50Hz.

- **Step-1>** Place the `CombineScript.py` and `MakeDataset.py` in the same folder that contains the UCI dataset. Ensure you have moved into the folder before running the scripts. If you are runing the scripts from a different folder, you will have to play around with the paths in the scripts to make it work.
- **Step-2>** Run `CombineScript.py` and provide the paths to test and train folders in UCI dataset. This will create a folder called `Combined` which will contain all the data from all the participants. This is how most of the datasets are organized. You may encounter similar dataset structures in the future.
- **Step-3>** Run `MakeDataset.py` and provide the path to `Combined` folder. This will create a Dataset which will contain the train, test and validation set. You can use this dataset to train your models.

Questions/Tasks

1. Plot the waveform for data from each activity class. Are you able to see any difference/similarities between the activities? You can plot a subplot having 6 colunms to show differences/similarities between the activities. Do you think the model will be able to classify the activities based on the data? **[1 marks]**

2. Do you think we need a machine learning model to differentiate between static activities (laying, sitting, standing) and dynamic activities(walking, walking_downstairs, walking_upstairs)? Look at the linear acceleration $(acc_x^2+acc_y^2+acc_z^2)$ for each activity and justify your answer. **[1 mark]**

3. Train Decision Tree using trainset and report Accuracy and confusion matrix using testset. **[1 mark]**

4. Train Decision Tree with varrying depths (2-8) using trainset and report accuracy and confusion matrix using Test set. Does the accuracy changes when the depth is increased? Plot the accuracies and reason why such a result has been obtained. **[1 mark]**

5. Use PCA (Principal Component Analysis) on Total Acceleration $(acc_x^2+acc_y^2+acc_z^2)$ to compress the acceleration timeseries into two features and plot a scatter plot to visualize different class of activities. Next, use TSFEL (a featurizer library) to create features (your choice which ones you feel are useful) and then perform PCA to obtain two features. Plot a scatter plot to visualize different class of activities. Are you able to see any difference? **[2 marks]**

6. Use the features obtained from TSFEL and train a Decision Tree. Report the accuracy and confusion matrix using test set. Does featurizing works better than using the raw data? Train Decision Tree with varrying depths (2-8) and compare the accuracies obtained in Q4 with the accuracies obtained using featured trainset. Plot the accuracies obtained in Q4 against the accuracies obtained in this question. **[2 marks]**

7. Are there any participants/ activitivies where the Model performace is bad? If Yes, Why? **[1 mark]**

Deployment! [4 marks]

For this exercise marks will not depend on what numbers you get but on the process you followed Utilize apps like `Physics Toolbox Suite` from your smartphone to collect your data in .csv/.txt format. Ensure at least 15 seconds of data is collected, trimming edges to obtain 10 seconds of relevant data. Collect 3-5 samples per activity class and report accuracy using both featurized and raw data. You have to train on UCI dataset (You can use the entire dataset if you want) and test it on the data that you have collected and report the accuracy and confusion matrix. Test your model's performance on the collected data, explaining why it succeeded or failed.

Things to take care of:

- Ensure the phone is placed in the same position for all the activities.
- Ensure the phone is in the same alignment during the activity as changing the alignment will change the data collected and will affect the model's performance.
- Ensure to have atleast 10s of data per file for training. As the data is collected at 50Hz, you will have 500 data samples.

NOTE:-

1. Show your results in a Jupyter Notebook or an MD file. If you opt for using an MD file, you should also include the code.
2. If you wish you can use the scikit-learn implementation of Decision Tree for the Mini-Project.