showcase_kaggleData

March 6, 2016

1 Showcase of kaggleData.py

As the title says, this is only a showcase.
For actual documentation, refer to kaggleData.py

```
In [1]: import kaggleData as kD
```

Most important for working with the original Kaggle datasets is recreating them.
Keep in mind, that the file `atlas-higgs-challenge-2014-v2.csv` is needed in the data folder.
First, we extract the whole csv file into a numpy array.

```
In [2]: csv_data,csv_header = kD.csvToArray()
```

We are able to extract all information we want form this data, this includes the data necessary to reproduce the AMS rating performed by Kaggle. The data sets we extract are * training set * test set * solution set

```
In [3]: train_data,train_header,test_data,test_header=kD.getOriginalKaggleSets(csv_data,csv_header)
        sol_data,sol_header = kD.getSolutionKey(csv_data,csv_header)
```

We need to prepare the sets further to avoid errors in classification. For most, that means we need to cut certain arrays and translate data into other datatypes.

```
In [4]: train_all = train_data[:,1:-2].astype(float)
        train_labels = kD.translateLabels(train_data[:,-1],["Label"]).astype(float)
        train_weights = train_data[:,-2].astype(float)
        test_all = test_data[:,1:].astype(float)
        header_all = test_header[1:]
```

We can extract single and multiple features as array. For instance we need to save the list of event IDs of the test dataset.

```
In [5]: test_events = kD._extractFeature("EventId",test_data,csv_header).astype(float)
```

We extract multiple features as we create feature subsets, like listed in Tab. 3 of the thesis.

```
In [6]: header_1 = ["DER_mass_MMC",
                   "DER_mass_transverse_met_lep",
                   "DER_mass_vis",
                   "DER_met_phi_centrality",
                   "DER_pt_ratio_lep_tau",
                   "PRI_tau_pt",
                   "DER_pt_h"]
        train_1 = kD._extractFeatures(header_1,train_data,train_header).astype(float)
        test_1 = kD._extractFeatures(header_1,test_data,test_header).astype(float)
```

For convinience, arrays for a feature set can be received directly.

```
In [7]: header_2,train_2,test_2 = kD.getFeatureSubset(train_data,test_data,train_header,test_header,2)
```

We can easily access the Kaggle leaderboard.

```
In [8]: pubLB,privLB = kD.getLeaderBoards()
```

The leaderboard data is saved in the order [user ID, score, rank] and sorted with respect to the ranks.
For comparing private and public leaderboards effectively, the data needs to be sorted w.r.t. the user IDs.

```
In [9]: import toolbox as tb
        sortedPriv=tb.sortByColumn(privLB,0)
        sortedPub=tb.sortByColumn(pubLB,0)
```

Further, we can access any Kaggle leaderboard.

```
In [10]: pubLB,privLB = kD.getLeaderBoards("https://www.kaggle.com/c/flavours-of-physics/leaderboard")
```

If we use an url of a running competition, we still receive data of the public leaderboard.

```
In [11]: pubLB,privLB = kD.getLeaderBoards("https://www.kaggle.com/c/home-depot-product-search-relevanc

Leaderboard-URL not found.
Private Leaderboard has not been found, is the competition still running?
Use getLeaderBoard(url) for a single leaderboard.
```

```
In [12]: pubLB,privLB

Out[12]: (array([[  2.66169000e+05,   4.48480000e-01,   1.00000000e+00],
                 [  2.65984000e+05,   4.49040000e-01,   2.00000000e+00],
                 [  2.67747000e+05,   4.50310000e-01,   3.00000000e+00],
                 ...,
                 [  2.82540000e+05,   1.47868000e+00,   1.33300000e+03],
                 [  2.83664000e+05,   1.47868000e+00,   1.33400000e+03],
                 [  2.69043000e+05,   1.47872000e+00,   1.33500000e+03]]), None)
```