Projection Pipelines

Eugene Choi

I. Purpose

The purpose of the Projection Pipelines project is to compare and contrast the results of multiple combinations of different projection methods and estimator algorithms. The metrics measured are accuracy score, precision score, recall score, time taken to train the pipeline, and the area under the receiver operating characteristics(ROC) curve. This information is recorded on a .csv file for easy access and legibility. In order to generate, fit, and gain results from multiple pipelines efficiently, pipeline_generator.py is used.

In prj_pipelines.py, PCA, IncrementalPCA, SparsePCA, FastICA, TruncatedSVD, and KMeans are used as projection methods and are combined with RandomForest, ExtraTrees, GradientBoosting, LinearSVC, SVC, NuSVC, SGDClassifier, KNeighbors, and RadiusNeighbors classifiers.

II. Pipeline Generator

The pipeline_generator.py program consists of a class called pipelines. An object of this class holds a list of pipelines to be trained, parameters to be tracked (for comparison), names of dataset files and the output file, necessary name and values of the target feature, and the header for labeling the .csv output file.

There are two functions to process the given datasets: data_process_sep and data_process. The former is used for datasets that are already separated into train and test sets while the latter is used for datasets that are not yet separated. The data_process_sep function combines the separate datasets for the total dataset cross validation step (explained further in the train_pipes). The data_process function separates the already combined dataset into test and train sets, with the test set size being 25% of the whole dataset. Both functions take the name of the target feature (*target*), the target value of the target feature (*target_val*), and the values that are not the target value (*non_target_val*). In the example given in prj_pipelines.py, the abalone dataset from the UCI repository is used. The target feature is sex, the target value is female, and the non-target values are male and infant (undeveloped). The header argument is used to define whether the input .csv files have a header labeling its features.

The create_pipeline function takes in the projection method (*PCA*, because it was the first method used before other methods were added) and the estimator (*estimator*). The function then creates names for each step of the pipeline ([*PCA name, estimator_(base_estimator)*]). It creates a list of steps which is used to create the pipeline. The newly created pipeline is appended to the list of pipelines of the class object.

Given a list of desired parameters to target (*target_params*), the add_target_params function adds each target to its params list. In order to target a parameter of a specific estimator, add the target as *estimator__parameter* to the input list.

The print_pipes function uses print_pipe to print out all the pipelines in the class object's pipelines list. The print_pipe function gets the parameters of the given pipe (*pipe*) and gets the names of the projection method and the estimator to print out.

The train_pipes function goes through the class object's list of pipelines and fits the pipeline, cross-validates the train dataset, and cross-validates the whole dataset (for comparison to make sure it is somewhat similar to the train dataset's results). Each method is timed and the accuracy, precision, recall scores as well as the area under the ROC curve are calculated. The result function is then called to create the string that contains the results to be written to the output file.

The result function takes in the pipe, name of the method used (pipeline only, cross-validation, or cross-validation on total dataset), time taken to train, the area under the ROC curve, and accuracy, recall, and precision scores. It also internally goes through the target parameters to add to the result string.

The create_header function creates the header that labels the output file.

The write_to_file function writes the given result string to the output file.

The create_file function creates the output file given the name of the file.

**NOTE: must be used with python3 - the time module will not work with python2.

III. Prj_Pipeline

prj_pipeline.py first imports all from pipelines generator. It then specifies the name of the desired dataset file and the desired output file name. The pipelines class object named pipe is then created and given the file names. The dataset is then processed with data_process.

```
from pipelines_generator import *

dataname = 'abalone.csv'
filename = 'abalone_test_combination.csv'

pipe = pipelines(dataname, filename)
pipe.data_process(0, 'F', ['M', 'I'], header="None")
```

Then the projection methods and the estimators to be used are put in a list.

```
prj_list = [PCA(), IncrementalPCA(), SparsePCA(), FastICA(), TruncatedSVD(), KMeans()]
alg_list = [RandomForestClassifier(), ExtraTreesClassifier(), GradientBoostingClassifier(),
```

(The alg_list may contain ensemble estimators that can take in base estimators. However, because the program was killed by the OOM killer when BaggingClassifier and AdaBoostClassifier were added, they were taken out for this example.)

For every estimator in alg_list, a pipeline is created with every projection method in prj_list.

```
for alg in alg_list:
    alg_name = str(alg).split("(")[0]
    print('\n *---------------- {} ----------------*'.format(alg_name))
    for prj in prj_list:
        pipe.create_pipeline(prj, alg)
```

PCA___n_components is added as the target parameter. The output file is created, and the pipelines are trained and cross-validated.

```
target_params = ['PCA__n_components']
pipe.add_target_params(target_params)
pipe.create_file()
pipe.train_pipes()
```

IV. Conclusion

Moving forward, the project may need an in-depth investigation on working with ensemble estimators such as the AdaBoostClassifier or the BaggingClassifier. Adding more methods projection or estimators would just mean importing those modules from sklearn in the pipeline generator. The pipeline generator can definitely also be used for more diverse combinations and metrics.