A Assigning Infill

Each transformation category has some default infill convention to serve as precursor to ML infill. For cases where a user wishes to override defaults assignments can be passed in the `assigninfill` parameter. ML infill is applied for columns not otherwise assigned, which default can be deactivated with the `MLinfill` parameter. When `MLinfill` is deactivated, columns not explicitly assigned will have infill per the default initialization associated with a transformation category. Here we demonstrate deactivating the ML infill default and assigning infill types zero infill to column1 and ML infill just to column2.

Not shown, if the data includes a label set or other features such as an index column appropriate for exclusion from ML infill basis, they should be designated with `labels_column` or `trainID_column`.

```
assigninfill = {'zeroinfill' : ['column1'],
                'MLinfill'   : ['column2'] }

train, train_ID, labels, \
val, val_ID, val_labels, \
test, test_ID, test_labels, \
postprocess_dict = \
am.automunge(df_train,
             MLinfill = False,
             assigninfill = assigninfill)
```

Note that the column headers can be assigned in `assigninfill` using the received column headers to apply consistent infill to all sets derived from an input column, or may alternatively be assigned using the returned column headers with transformation suffix appenders to assign infill to distinct returned columns, which take precedence.

B ML Infill Parameters

The default ML infill architecture is a Scikit-Learn random forest with default parameters. Alternate auto ML options are currently available as CatBoost, FLAML, and AutoGluon. Parameters can be passed to the models with `ML_cmnd`.

First we'll demonstrate applying ML infill with the CatBoost library. Note that we can either defer to the library default parameters or also pass parameters to the model initializations or fit operations. Here we also demonstrate assigning a particular GPU device number.

```
ML_cmnd = {'autoML_type' : 'catboost'}

#GPU device assignment takes place in model initialization
ML_cmnd.update({'MLinfill_cmnd' :
                {'catboost_classifier_model'   :
                  {'task_type' : 'GPU', 'devices' : 0 },
                 'catboost_regressor_model'    :
                  {'task_type' : 'GPU', 'devices' : 0 }}})

train, train_ID, labels, \
val, val_ID, val_labels, \
test, test_ID, test_labels, \
postprocess_dict = \
am.automunge(df_train,
             MLinfill = True,
             ML_cmnd = ML_cmnd)
```

The FLAML library is also available. Here we demonstrate setting a training time budget (in seconds) for each imputation model.

```
ML_cmnd = {'autoML_type' : 'flaml'}
```

```
ML_cmnd.update({'MLinfill_cmnd' :
                 {'flaml_classifier_fit' :  {'time_budget' : 60 },
                  'flaml_regressor_fit' : {'time_budget' : 60 }}})

train, train_ID, labels, \
val, val_ID, val_labels, \
test, test_ID, test_labels, \
postprocess_dict = \
am.automunge(df_train,
             MLinfill = True,
             ML_cmnd = ML_cmnd)
```

As another demonstration, here is an example of applying the AutoGluon library for ML infill and also applying the `best_quality` option which causes AutoGluon to train extra models for the aggregated ensembles. (Note this will likely result in large disk space usage, especially when applying to every column, so recommend saving this for final production if at all.)

```
ML_cmnd = {'autoML_type' : 'autogluon'}

ML_cmnd.update({'MLinfill_cmnd' :
                 {'AutoGluon' :
                   {'presets' : 'best_quality' }}})

train, train_ID, labels, \
val, val_ID, val_labels, \
test, test_ID, test_labels, \
postprocess_dict = \
am.automunge(df_train,
             MLinfill = True,
             ML_cmnd = ML_cmnd)
```

To be complete, here we'll demonstrate passing parameters to the Scikit-Learn random forest models. Note that for random forest there are built in methods to perform grid search or random search hyperparameter tuning when parameters are passed as lists or distributions instead of static figures. Here we'll demonstrate performing tuning of the `n_estimators` parameter (which otherwise would default to 100).

```
ML_cmnd = {'autoML_type' : 'randomforest'}

#by passing random forest parameters as a list
#each imputation model will perform a grid search

ML_cmnd.update({'MLinfill_cmnd' :
                 {'RandomForestClassifier' :
                   {'n_estimators' : [100, 222, 444] },
                  'RandomForestRegressor' :
                   {'n_estimators' : [100, 222, 444] }}})

train, train_ID, labels, \
val, val_ID, val_labels, \
test, test_ID, test_labels, \
postprocess_dict = \
am.automunge(df_train,
             MLinfill = True,
             ML_cmnd = ML_cmnd)
```

C Broader Impacts

The following discussions are somewhat speculative in nature. At the time of this writing Automunge has yet to establish what we would consider a substantial user base and there may be a bias towards optimism at play in how we have been proceeding, which we believe is our sole leverage of bias.

From an ethical standpoint, we believe the potential benefits of our platform far outweigh any negative aspects. We have sought to optimize the postmunge(.) function for speed, used as a proxy for computational efficiency and carbon intensity. As a rule of thumb, processing times for equivalent data in the postmunge(.) function, such as could be applied to streams of data in inference, have shown to operate on the order of twice the speed of initial preparations in the automunge(.) function, although for some specific transforms like those implementing string parsing that advantage may be considerably higher. While the overhead may prevent achieving the speed of directly applying manually specified transformations to a dataframe, the postmunge(.) speed gets close to manual transformations with increasing data size.

We believe too that the impact to the machine learning community of a formalized open source standard to tabular data preprocessing could have material benefits to ensuring reproducibility of results. There for some time has been a gap between the wide range of open source frameworks for training neural networks in comparison to options for prerequisites of data pipelines. I found some validation for this point from the tone of the audience Q&A at a certain 2019 NeurIPS keynote presentation by the founder of a commercial data wrangling package. In fact it may be considered a potential negative impact of this research in the risk to commercial models of such vendors, as Automunge's GNU GPL 3.0 license coupled with patent pending status on the various inventions behind our library (including parsed categoric encodings, family tree primitives, ML infill, and etc.) will preclude commercial platforms offering comparable functionality. We expect that the benefits to the machine learning community in aggregate will far outweigh the potential commercial impacts to this narrow segment. Further, benefits of automating machine learning derived infill to missing data may result in a material impact to the mainstream data science workflow. That old rule of thumb often thrown around about how 80% of a machine learning project is cleaning the data may need to be revised to a lower figure.

Regarding consequence of system failure, it should be noted that Automunge is an industry agnostic toolset, with intention to establish users across a wide array of tabular data domains, potentially ranging from the trivial to mission critical. We recognize that with this exposure comes additional scrutiny and responsibility. Our development has been performed by a professional engineer and we have sought to approach validations, which has been an ongoing process, with a commensurate degree of rigor.

Our development has followed an incremental and one might say evolutionary approach to systems engineering, with frequent and sequential updates as we iteratively added functionality and transforms to the library within defined boundaries of the data science workflow. The intent has always been to transition to a more measured pace at such time as we may establish a more substantial user base.

D Intellectual Property Disclaimer