Lab Data Preparation

Summary

In this lab you will:

- Try different strategies in the notebook for data preprocessing and observe changes in the pipeline performance

Refer to the demo videos from this lesson for a step-by-step demonstration of how to complete the lab.

Instructions

1. Save original notebook version for the banknote_auth experiment
    a. Open the AutoAI-generated notebook for the banknote_auth experiment
    b. Select File -> Save Version
2. Retrieve the Data Profile
    a. In a different browser tab from the notebook, navigate to the Data Profile for the banknote_auth experiment
    b. Observe that all of the attributes are numerical for this problem
    c. Recall that numerical attributes are not scaled by default for an AutoAI experiment. How do you think this might have affected the outcome of the experiment?
3. Review the Compose Pipeline cell of the notebook
    a. Find the code block under the comment # composing steps for preprocessor Pipeline
    b. Find the line that appends num_scaler to the pipeline using the autoai_libs OptStandardScaler
    c. Review the parameters to the OptStandardScaler from IBM's documentation: (also see next page) https://dataplatform.cloud.ibm.com/docs/content/wsj/analyze-data/autoai-lib-python.html
    d. Set the parameters `num_scaler_with_mean`, `num_scaler_with_std`, and `use_scaler_flag` to `True`
    e. Re-run the notebook. Does it affect the output scores for holdout and cross-validation?
        i. Recall that the pipeline was optimized with the given inputs and we've now changed the inputs. We should be aware of this as we continue to make changes to the notebook. Performing optimization with these new inputs could give us different results.
4. Save notebook and revert to previously saved version of notebook
    a. File -> Save Version
    b. File -> Revert to Version and select your previously saved version
    c. Click Revert

Option	Description
num_scaler_copy	boolean, optional, default True. If False, try to avoid a copy and do in-place scaling instead. This is not guaranteed to always work. With in-place, for example, if the data is not a NumPy array or scipy.sparse CSR matrix, a copy may still be returned.
num_scaler_with_mean	boolean, True by default. If True, center the data before scaling. This does not work (and will raise an exception) when attempted on sparse matrices, because centering them entails building a dense matrix which in common use cases is likely to be too large to fit in memory.
num_scaler_with_std	boolean, True by default. If True, scale the data to unit variance (or equivalently, unit standard deviation).
use_scaler_flag	boolean, flag that indicates that this transformer will be active. If False, transform(X) outputs the input numpy array X unmodified. Default is True.

5. Optional: Repeat for the parkinsons_updrs experiment

    a. Follow the above instructions to save the original version of the notebook and retrieve the data Preview and Profile
    b. Observe that there is a mix of categorical and numerical attributes for this dataset
        i. Recall that the AutoAI default for categorical variables is ordinal encoding, meaning that the sex column with its values '0' for male and '1' for female has an ordinal encoding. In the next step, you'll use one-hot encoding for the sex attribute instead.
        ii. As above with the banknote_auth experiment, the numerical attributes have not been standardized or scaled by default. You'll experiment with scaling and standardizing them in the next step.
    c. One hot encode the sex column
        i. Within the 5. Preprocess Data cell under the code block with comment `# composing steps for preprocessor_features_categorical Pipeline`, find the line of code that appends `cat_encoder` using the `autoai_libs CatEncoder`.
        ii. See below or view the documentation of `CatEncoder`
        iii. Change the argument for `encoding` from `ordinal` to `one-hot`. Since there is only one encoding style set for all of the categorical variables, this is also going to one-hot encode the age column.
        iv. Re-run the notebook and observe how this changes the holdout and cross-validation scores.
    d. Scale/standardize numerical attributes

i. Repeat the steps in 3 above to standardize and scale the numerical attributes.
ii. Re-run the notebook. How does this affect the holdout and cross-validation scores?

e. Save notebook version

autoai_libs.transformers.exportable.CatEncoder()

This is a wrapper for categorical encoder. If encoding parameter is 'ordinal', internally it currently uses sklearn OrdinalEncoder. If encoding parameter is 'onehot', or 'onehot-dense' internally it currently uses sklearn OneHotEncoder

Usage:

```
autoai_libs.transformers.exportable.CatEncoder(encoding, categories, dtype, handle_unknown, sklearn_version_fam
```

Option	Description
`encoding`	str, 'onehot', 'onehot-dense' or 'ordinal'. The type of encoding to use (default is 'ordinal') 'onehot': encode the features using a one-hot aka one-of-K scheme (or also called 'dummy' encoding). This creates a binary column for each category and returns a sparse matrix. 'onehot-dense': the same as 'onehot' but returns a dense array instead of a sparse matrix. 'ordinal': encode the features as ordinal integers. This results in a single column of integers (0 to n_categories - 1) per feature.
`categories`	'auto' or a list of lists/arrays of values. Categories (unique values) per feature: 'auto' : Determine categories automatically from the training data. `list` : `categories[i]` holds the categories expected in the ith column. The passed categories must be sorted and should not mix strings and numeric values. The used categories can be found in the `encoder.categories_` attribute.
`dtype`	number type, default np.float64 Desired dtype of output.
`handle_unknown`	'error' (default) or 'ignore'. Whether to raise an error or ignore if a unknown categorical feature is present during transform (default is to raise). When this parameter is set to 'ignore' and an unknown category is encountered during transform, the resulting one-hot encoded columns for this feature will be all zeros. In the inverse transform, an unknown category will be denoted as None. Ignoring unknown categories is not supported for `encoding='ordinal'`.
`sklearn_version_family`	str indicating the sklearn version for backward compatibiity with versions 019, and 020dev. Currently unused. Default is None.
`activate_flag`	flag that indicates that this transformer will be active. If False, transform(X) outputs the input numpy array X unmodified.