AWS Deployment | schwartz March 12, 2020

1 Pipeline

Best `KFoldsGridSearch(RF(X.OneHot.NAflag.medianImpute,Y,pars), parsGrid)` performed as follows:

	Final Hold Out (X^*,Y^*)	*K-Folds Test/Train*	$\pm\ \hat{\sigma}$
Recall	0.922	0.946/0.959	± 0.0200/0.0045
Precision	0.0359	0.0366/0.0372	± 0.00066/0.00030
Accuracy	0.649	0.640/0.641	± 0.0023/0.0020

suggesting limited over-fitting/inappropriate model complexity with good out of sample generalizability. Model under-fitting/score improvement was not pursued as random forests are typically decent performers.

- Class imbalance was addressed by setting the classification threshold at the observed default rate $\overline{Y}$.
- Best model was taken to be the one that optimized *recall score* over `parsGrid` (at the $\overline{Y}$ threshold).
- Final *sensitivity-specificity* tuning via classification threshold selection was based on a cost-benefit analysis using estimations of (a) potential customer loss value and (b) expected losses upon default as a result of actions resulting from a 'default' prediction (relative to the 'status quo' under inaction); which provided the observed tradeoff in (X^*,Y^*): {*Recall*: 0.775, *Precision*: 0.0596, *Accuracy* 0.830}

2 Quality

- `sklearn.ensemble.IsolationForest` did not indicate specific data points as excessively anomalous.
  - I opted not to put effort into checks for problematic systematic data patterns or other QC tasks.
- *Propensity score* and *KS-test covariate balance* analyses did not identify feature divergences from the labelled to unlabelled data, so extrapolation of predictions to the unlabeled data appeared justifiable.
  - For these analyses features suffering from high multicollinearity were identified and removed via `statsmodels.stats.outliers_influence.variance_inflation_factor`

3 Usage

For your convenience, generated example API calls are available at https://3.94.204.113:5000/generate and visiting https://3.94.204.113:5000/default?<var_1>=<val_1>&..&<var_p>=<val_p> returns:

```
{ 'Default_Prediction': <0 or 1, 1 means default is predicted>,
  'Default_Probability': <probabilistic risk of default, 1 the highest risk>,
  'Relative_Risk_Ratio': <Default_Probability ÷ Population_Default_Probability>,
  'Extrapolation_Percentile': <percentile of 'degree of extrapolation', 100 the worst>,
  'Cost_of_Actualized_False_Positive: <estimated financial cost, in currency>,
  'Cost_of_Actualized_False_Negative': <estimated financial cost, in currency>,
  'Long_Run_False_Positive_Cost': <Cost_Of_Actualized_FP × Non_Default_Probability>,
  'Long_Run_False_Negative_Cost': <Cost_Of_Actualized_FN × Default_Probability>       }
```

where `<var_j>` names a variable and `<var_j>` is it's value (converted to a number if possible):

- if `<var_j>` does not appear as a column in `X`, both `<var_j>` and `<val_j>` will be ignored
- if some column `<var_j>` in `X` used in model fitting is not named, then `<val_j>` is taken as NA
- if `<val_j>` did not appear in column `<var_j>` of `X` used in model fitting, `<val_j>` is taken as NA

Large `Extrapolation_Percentile` values are suggestive of potential untrustworthiness of the model for this specific prediction.