Generali Data Challenge: Churn Prediction

The reasoning followed in solving the problem is divided into two parts: the first part is about exploration and preprocessing of the dataset; the second part is about choosing the basic model and fine tuning of the hyperparameters.

- Exploration and preprocessing of the dataset

While I was looking at both train and test set, I notice that in both datasets there are many NaNs (both as np.nan and as '#' character), features with a single value and categorical features. Initially I proceed to replace the placeholder '#' with np.nan, then I check how many NaNs or None values are present. Features such as 'feature_36 / 37/38/39' contain about 8500 NaNs, while features 'feature_66 / 67/68 / 70-81' and 'feature_245 / 247-249' contain from 4000 to 5000 NaNs. I drop columns with more than 4000 NaNs from both datasets. Furthermore, I also drop features with a single value (all 0 or 1) since for a potential predictive model this type of feature is equivalent to adding noise and it can get worse the prediction.
I check which features contain numeric and non-numeric values. The features 'feature_6/13/14' contain numbers in string format and some strings not corresponding to a number (for example 'GE'), so I proceed to convert the strings with their respective numerical values and convert the other strings not attributable to a number with np.nan. This way my datasets contain only numeric values.
As a method of feature selection and dimensionality reduction, I have chosen to use the correlation matrix and to eliminate a feature of the pair of features with correlation >=0.75. I also tried to use other methods such as PCA (and its variants) and Factor Analysis but in the validation phase, the datasets built with these two methods which were then used to train the models led to worse results than those with dataset obtained simply by eliminating redundant features.
The NaN values have not been imputed because I believe that in this case they contain latent information that could be useful for the algorithm in discovering some patterns in the data.
Finally, since it is a binary classification problem, I checked if there was no unbalanced class in the target. There are 8772 class 0 observations and 1228 class 1 observations, so we are dealing with a case of imbalanced dataset that will then be treated differently, based on the choice of the algorithm.
The train and test set that will be used to train (train set) and predict (test set) contain a total of 101 features.

- Model selection and fine tuning of the hyperparameters

The choice concerned two non-linear models that have been winners of some competitions on Kaggle with tabular data: the Light Gradient Boosted Machine (LGBM) and the Extreme Gradient Boosting (XGBoost). Since these are two models based on decision trees, it was not necessary to standardize or normalize the data because this has no influence in the process of creating the tree. Regarding the problem of imbalanced dataset, this was solved by setting the parameter 'scale_pos_weight' to 7.14 (= 8772/1228) for both models, without the need to use undersampling and/or oversampling algorithms (e.g. RandomUndersampler and/or SMOTE). The parameter 'objective' for both LGBM and XGBoost is respectively 'binary' and 'binary:logistic'. The parameter 'metric' is

both set to 'binary_logloss'. The train set was split into train_X set and validation set with validation set size = 0.25. In the training phase of both models, the train_X set was used to fit the model and the validation set to calculate the F1 score.
For the choice and the fine tuning of the hyperparameters, I used the Optuna library. Early stopping was set to 100 and 5000 trials were carried out for each model.
The complete list of optimized hyperparameters is in the following table. The choice of the model to use for the prediction is simply the one that achieves the highest F1 score in the validation set.

LGBM

boosting: dart
num_leaves: 27
learning_rate: 0.01
n_estimators: 290
min_child_samples: 52
colsample_bytree: 0.6
reg_alpha: 2.121067575840617
bagging_fraction: 0.6672508287937184
lambda_l2: 0.000246505281239431
min_data_in_leaf: 11
max_delta_step: 4
drop_rate: 0.41966285307038387
min_data_in_bin: 21
lambda_l1: 0.0012595398095782777
feature_fraction: 0.6083195334376993
bagging_freq: 1

Value F1 : 0.3671875

XGBoost

booster: gbtree
eta: 7.961047489659033e-07
max_depth: 4
n_estimator: 430
min_child_weight: 10
gamma: 2.247424300918677e-06
alpha: 2.8228365845205365e-05
max_delta_step: 8
subsample: 0.9
colsample_bytree: 0.5
colsample_bylevel: 0.7000000000000001
colsample_bynode: 0.65
lambda: 0.00044868464635031266
grow_policy: lossguide

Value F1 : 0.3485554520037278

The LGBM model has performed better, so the choice of the final model falls on the latter. The file containing the prediction derive from the LGBM model with the hyperparameters described and shown previously.