Generali Data Challenge: Churn Prediction

Il ragionamento seguito nella risoluzione generale del problema è suddiviso in due parti: una prima parte di esplorazione e aggiustamento del dataset; una seconda parte di scelta del modello base e ottimizzazione degli iperparametri.

- Esplorazione e preprocessamento del dataset

Esplorando sia il train sia il test set, noto che in entrambi i dataset sono presenti molti NaN (sia come np.nan che col carattere #), features con un solo valore e features categoriali. Inizialmente procedo a sostituire il carattere # con np.nan, poi controllo quanti valori NaN o None sono presenti. Features quali 'feature_36/37/38/39' contengono circa 8500 NaN, mentre le features 'feature_66/67/68/70-81' e 'feature_245/247-249' contengono dalle 4000 a 5000 NaN. Elimino da entrambi i dataset le colonne con più di 4000 NaN. Inoltre elimino anche le features con un solo valore (tutti 0 o 1) dato che per un eventuale modello predittivo questo tipo di feature equivale ad aggiungere solo rumore e può peggiorare la previsione.
Controllo quali features contengono valori numerici e non. Le features 'feature_6/13/14' contengono numeri in formato stringa e qualche stringa non riconducibile a un numero (ad esempio 'GE'), perciò procedo a convertire le stringhe con i loro rispettivi valori numerici e converto le altre stringhe non riconducibili a un numero con np.nan. In questo modo i miei dataset contengono solo valori numerici.
Come metodo di feature selection e dimensionality reduction ho scelto di utilizzare la matrice di correlazione e di eliminare una feature della coppia di features con correlazione >=0.75. Ho provato ad utilizzare anche altri metodi come la PCA (e le sue varianti) e Factor Analysis ma in fase di validation, i dataset costruiti con questi due metodi che sono stati poi usati per addestrare i modelli hanno portato a risultati peggiori rispetto a quelli col dataset ricavato semplicemente eliminando le features ridondanti.
I valori NaN non sono stati imputati perché credo che in questo caso contengano un informazione latente che potrebbe essere utile per l'algoritmo nel scoprire alcuni pattern nei dati.
Infine, trattandosi di un problema di classificazione binaria, ho controllato se nel target non ci fosse una classe sbilanciata. Ci sono 8772 osservazioni di classe 0 e 1228 osservazione di classe 1, quindi ci troviamo davanti a un caso di imbalanced dataset che verrà poi trattato in base alla scelta dell'algoritmo.
Il train e test set che sarà utilizzato per fare l'addestramento(train set) e previsione(test set) contengono in totale 101 features.

- Scelta del modello e fine tuning degli iperparametri

La scelta è caduta su due modelli non lineari che sono risultati vincitori di alcune competizioni su Kaggle con dati tabulari: il Light Gradient Boosted Machine (LGBM) e l' Extreme Gradient Boosting (XGBoost). Trattandosi di due modelli basati sugli alberi

decisionali, non è stato necessario standardizzare o normalizzare i dati perché ciò non ha alcuna influenza nel processo di creazione dell'albero. Per quanto riguarda il problema dell'imbalance dataset, questo è stato risolto impostando il parametro 'scale_pos_weight' a 7.14 (= 8772/1228) ad entrambi i modelli, senza la necessità di dover usare algoritmi di undersampling e/o oversampling (ad esempio RandomUndersampler e/o SMOTE). Il parametro objective sia per LGBM che per XGBoost è rispettivamente 'binary' e 'binary:logistic'. Il parametro 'metric' è in entrambi impostato a 'binary_logloss'. Il train set è stato diviso in train_X set e validation set con validation set size = 0.25. Nella fase di addestramento di entrambi i modelli è stato usato il train_X set per fittare il modello e il validation set per calcolare l' F1 score.
Per la scelta degli iperparametri per entrambi i modelli e per il fine tuning degli stessi, ho utilizzato la libreria Optuna. L'early stopping è stato impostato a 100 e sono state effettuate 5000 trials per ogni modello.
La lista completa degli iperparametri ottimizzati è nella seguente tabella. La scelta del modello da usare per la previsione è semplicemente quello che raggiunge F1 score più alto nel 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

Il modello LGBM ha performato meglio, dunque la scelta del modello finale ricade su quest'ultimo. Il file contenente le previsioni derivano dal modello LGBM con gli iperparametri descritti e mostrati precedentemente.