Churn Prediction Model Highest ROC AUC on Kaggle with Resampling & SMOTE-NC

YOUNES AISSAOUI

2023

Make sure you have all the packages and modules both in .R and Python

```
library(tidymodels)
library(tidyverse)
library(reticulate)
library(bonsai)
use_python("C:/Users/YOUNES/AppData/Local/Programs/Python/Python310/python.exe")
```

```
import pandas as pd
import numpy as np
import imblearn
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import RandomUnderSampler
from imblearn.over_sampling import SMOTENC
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
```

Importing Raw Data

```
df <- pd.read_csv("C:/Users/YOUNES/Desktop/Telco-Customer-Churn.csv")
```

Imbalanced data ?

```
df %>%  group_by(Churn) %>% summarise(count=n())
```

```
## # A tibble: 2 x 2
##   Churn count
##   <chr> <int>
## 1 No     5163
## 2 Yes    1869
```

Splitting Imbalanced Data

```
set.seed(252)
Imbalanced_data_split <- initial_split(df, prop = 0.8)
Imbalanced_data_train <- training(Imbalanced_data_split)
Imbalanced_data_test <- testing(Imbalanced_data_split)
```

Balancing data Using SMOTE-NC (Python Chunk) ** NO DATA LEAKAGE **

```
dfb = pd.read_csv("C:/Users/YOUNES/Desktop/Telco-Customer-Churn.csv")
dfb = dfb.iloc[:,1:]
X = dfb.iloc[:,:-1]
y = dfb.iloc[:,-1]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,random_state=252)
categorical_features = [0, 1, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]  # Example indices of ca
numerical_features = [4, 17, 18]
preprocessor = ColumnTransformer(
    transformers=[
        ('cat', OneHotEncoder(), categorical_features),
        ('num', StandardScaler(), numerical_features)
    ])
X_train_encoded = preprocessor.fit_transform(X_train)
X_test_encoded = preprocessor.fit_transform(X_test)
oversampler = SMOTENC(sampling_strategy=0.4, categorical_features=[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11
,k_neighbors=5,random_state=252)
undersampler = RandomUnderSampler(sampling_strategy=0.6,random_state=252)
X_oversampled, y_oversampled = oversampler.fit_resample(X_train_encoded, y_train)
X_resampled, y_resampled = undersampler.fit_resample(X_oversampled, y_oversampled)
cat_encoder = preprocessor.named_transformers_['cat']
cat_feature_names = cat_encoder.get_feature_names_out()
feature_names = list(cat_feature_names) +  ["tenure" ,"MonthlyCharges" ,"TotalCharges"]
df_resampled = pd.DataFrame(X_resampled, columns=feature_names)
df_resampled['Churn'] = y_resampled
df_test_encoded = pd.DataFrame(X_test_encoded, columns=feature_names)
X_train=df_resampled.iloc[:,:-1]
y_train=df_resampled.iloc[:,-1]
X_test = df_test_encoded
series_df = pd.DataFrame(y_test, columns=['Churn'])
series_df = series_df.reset_index(drop=True)
balanced_data_test = pd.concat([X_test, series_df], axis=1)
balanced_data_train = df_resampled
```

Splitting No data leakage

```
balanced_data_test = as.data.frame(py$balanced_data_test)
balanced_data_train = as.data.frame(py$balanced_data_train)
dfb <- rbind(balanced_data_train,balanced_data_test)
```

```
ind <- list(analysis = seq(nrow(balanced_data_train)), assessment = nrow(balanced_data_train) + seq(nrow
balanced_data_split <- make_splits(ind, dfb)
balanced_data_split
```

```
## <Analysis/Assess/Total>
## <4392/1407/5799>
```

Logistic regression,Random Forest and lightgbm all with the balanced data Set All with tuning. **18 minutes to tune on my machine**

```
Churn_metrics <-
  metric_set(roc_auc)
recipe_all <-
  recipe(Churn ~ ., data = balanced_data_train)
balanced_logreg_mod <-
  logistic_reg() %>%
  set_engine("glm")

balanced_tlogreg_mod <-
  logistic_reg(penalty = tune(),
               mixture = tune()) %>%
  set_engine("glmnet")

balanced_rf_mod <-
  rand_forest(trees = 1000,mtry=tune()) %>%
  set_engine("ranger") %>%
  set_mode("classification")

balanced_lgbm_mod <-
  boost_tree(
    mtry = tune(),
    trees = tune(),
    tree_depth = tune(),
    learn_rate = tune(),
    min_n = tune(),
    loss_reduction = tune()
  ) %>%
  set_engine(engine = "lightgbm") %>%
  set_mode(mode = "classification")

wf_set_tune <-
  workflow_set(
    list(plain = recipe_all),
    list(rf = balanced_rf_mod,
      lightgbm = balanced_lgbm_mod,
      logreg = balanced_logreg_mod,
      tlogreg = balanced_tlogreg_mod
     )
  )
set.seed(345)
Churn_folds <- vfold_cv(balanced_data_train, v = 3, strata = Churn)
```

```
tune_results <-
  workflow_map(
    wf_set_tune,
    "tune_grid",
    resamples = Churn_folds,
    grid = 10,
    metrics = Churn_metrics,
    verbose = TRUE
    )
rank_results(tune_results, rank_metric = "roc_auc") %>%
  select(-`.config`, -n,-preprocessor) %>%
  filter(.metric == "roc_auc")
```

```
## # A tibble: 31 x 6
##    wflow_id       .metric  mean std_err model         rank
##    <chr>          <chr>   <dbl>   <dbl> <chr>        <int>
##  1 plain_lightgbm roc_auc 0.845 0.00429 boost_tree       1
##  2 plain_tlogreg  roc_auc 0.844 0.00496 logistic_reg     2
##  3 plain_tlogreg  roc_auc 0.844 0.00494 logistic_reg     3
##  4 plain_tlogreg  roc_auc 0.843 0.00490 logistic_reg     4
##  5 plain_logreg   roc_auc 0.843 0.00492 logistic_reg     5
##  6 plain_lightgbm roc_auc 0.843 0.00383 boost_tree       6
##  7 plain_lightgbm roc_auc 0.843 0.00419 boost_tree       7
##  8 plain_rf       roc_auc 0.843 0.00419 rand_forest      8
##  9 plain_tlogreg  roc_auc 0.843 0.00444 logistic_reg     9
## 10 plain_tlogreg  roc_auc 0.843 0.00450 logistic_reg    10
## # ... with 21 more rows
```

Choosing the Top Two Models

lightgbm +0.868 ROC AUC

```
lightgbm_hyperparameters <- tune_results %>%
  extract_workflow_set_result("plain_lightgbm") %>%
  select_best(metric = "roc_auc")
lightgbm_validation_results <- tune_results %>%
  extract_workflow("plain_lightgbm") %>%
  finalize_workflow(lightgbm_hyperparameters) %>%
  last_fit(split =  balanced_data_split, metrics = Churn_metrics)
collect_metrics(lightgbm_validation_results)
```

```
## # A tibble: 1 x 4
##   .metric .estimator .estimate .config
##   <chr>   <chr>          <dbl> <chr>
## 1 roc_auc binary         0.868 Preprocessor1_Model1
```

Reg logistic Regression +0.872 ROC AUC

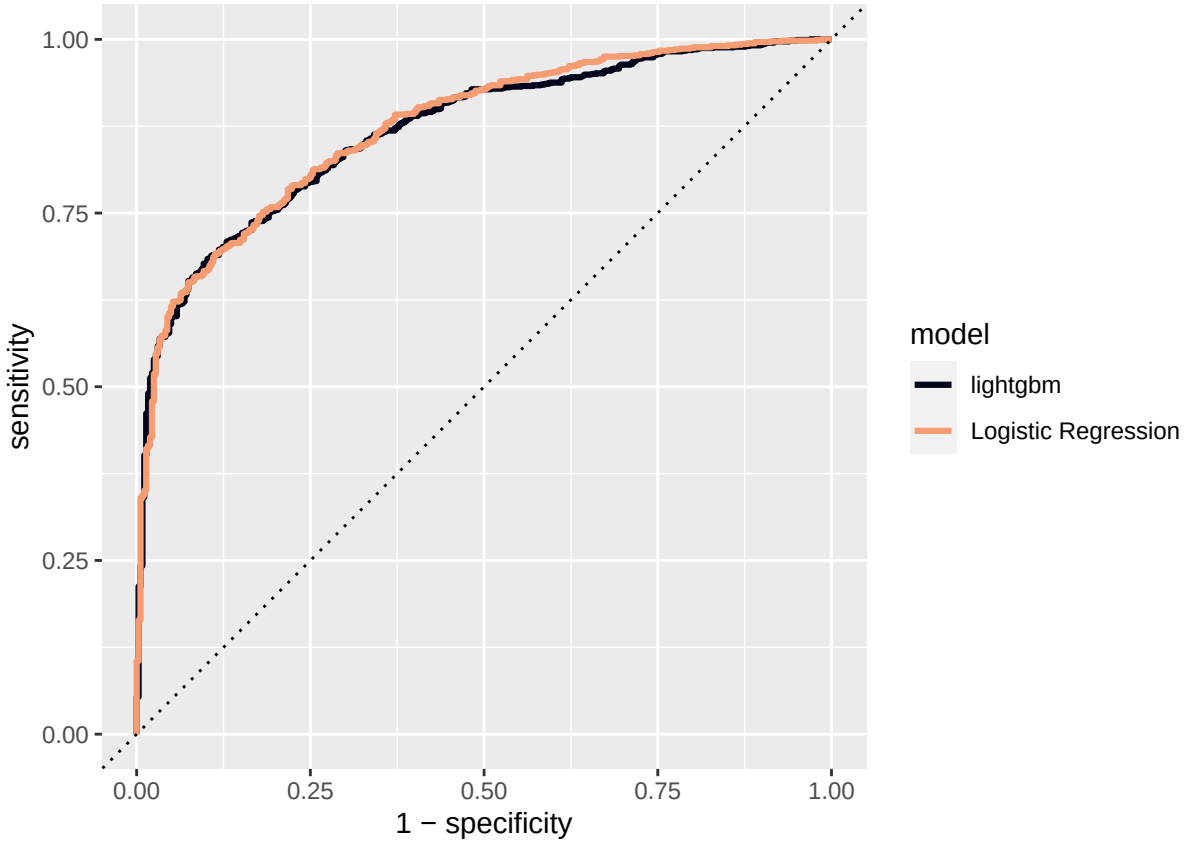
```
plain_tlogreg_hyperparameters <- tune_results %>%
  extract_workflow_set_result("plain_tlogreg") %>%
  select_best(metric = "roc_auc")
plain_tlogreg_validation_results <- tune_results %>%
  extract_workflow("plain_tlogreg") %>%
  finalize_workflow(plain_tlogreg_hyperparameters) %>%
  last_fit(split =  balanced_data_split, metrics = Churn_metrics)
collect_metrics(plain_tlogreg_validation_results)
```

```
## # A tibble: 1 x 4
##   .metric .estimator .estimate .config
##   <chr>   <chr>          <dbl> <chr>
## 1 roc_auc binary         0.873 Preprocessor1_Model1
```

Plotting the ROC Curve for both models

```
log_auc <-
  plain_tlogreg_validation_results %>%
  collect_predictions() %>%
  roc_curve(Churn, .pred_No) %>%
  mutate(model = "Logistic Regression")

gbm_auc <-
  lightgbm_validation_results %>%
  collect_predictions() %>%
  roc_curve(Churn, .pred_No) %>%
  mutate(model = "lightgbm")
bind_rows(log_auc,gbm_auc) %>%
  ggplot(aes(x = 1 - specificity, y = sensitivity, col = model)) +
  geom_path(linewidth = 1.1, alpha = 1.1) +
  geom_abline(lty = 3) +
  coord_equal() +
  scale_color_viridis_d(option = "rocket", end = 0.8)
```


1.00
0.75
0.50
0.25
0.00
sensitivity
0.00
0.25
0.50
0.75
1.00
1 − specificity
model
lightgbm
Logistic Regression