

FIN42110: HomeWork 8

Ross Kearney

28th April 2022

1 Original Cross Validated AUC

The AUC of the given Credit Card fraud dataset was initially equal to 0.779

```
[17] validation_0-aucpr:0.92844 validation_1-aucpr:0.77559
[18] validation_0-aucpr:0.93706 validation_1-aucpr:0.77107
[19] validation_0-aucpr:0.94220 validation_1-aucpr:0.76967
[20] validation_0-aucpr:0.94803 validation_1-aucpr:0.77165
[21] validation_0-aucpr:0.95508 validation_1-aucpr:0.77534
[22] validation_0-aucpr:0.95861 validation_1-aucpr:0.77568
[23] validation_0-aucpr:0.96240 validation_1-aucpr:0.77517
[24] validation_0-aucpr:0.96627 validation_1-aucpr:0.77667
[25] validation_0-aucpr:0.96931 validation_1-aucpr:0.77556
[26] validation_0-aucpr:0.97244 validation_1-aucpr:0.77636
[27] validation_0-aucpr:0.97556 validation_1-aucpr:0.77589
[28] validation_0-aucpr:0.98005 validation_1-aucpr:0.77770
[29] validation_0-aucpr:0.98467 validation_1-aucpr:0.77836
Training PR AUC:0.985
Test PR AUC:0.779
In [21]:
```

Figure 1: AUC from original code

2 AUC Improvement

In order to improve on the cross-validated AUC, the Synthetic Minority Over-sampling Technique was used.

In this dataset, only 492 out of 284807 data points contained fraudulent activity. This is a highly imbalanced data set, meaning there are too few examples of the minority class for a model to effectively learn the decision boundary.

To solve this problem oversampling of the minority class can be carried out, using the 'imblearn.SMOTE' package.

below is the improved cross-validated AUC, giving a test score of 0.974! A major improvement.

```
[17] validation_0-aucpr:0.98193 validation_1-aucpr:0.97690
[18] validation_0-aucpr:0.98350 validation_1-aucpr:0.97493
[19] validation_0-aucpr:0.98556 validation_1-aucpr:0.97516
[20] validation_0-aucpr:0.98595 validation_1-aucpr:0.97452
[21] validation_0-aucpr:0.99029 validation_1-aucpr:0.97460
[22] validation_0-aucpr:0.99110 validation_1-aucpr:0.97601
[23] validation_0-aucpr:0.99216 validation_1-aucpr:0.97460
[24] validation_0-aucpr:0.99234 validation_1-aucpr:0.97401
[25] validation_0-aucpr:0.99255 validation_1-aucpr:0.97431
[26] validation_0-aucpr:0.99305 validation_1-aucpr:0.97358
[27] validation_0-aucpr:0.99369 validation_1-aucpr:0.97428
[28] validation_0-aucpr:0.99495 validation_1-aucpr:0.97506
[29] validation_0-aucpr:0.99543 validation_1-aucpr:0.97426
Training PR AUC:0.995
Test PR AUC:0.974
In [22]:
```

Figure 2: Improved AUC

3 Python Code

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Thu Apr 28 17:54:41 2022

@author: rosskearney
"""

import numpy as np
import pandas as pd
from xgboost import XGBClassifier
from sklearn.metrics import average_precision_score
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from xgboost import plot_importance
from imblearn.over_sampling import SMOTE

from scipy import stats

# zscore function
def z_score(df): return (df-df.mean())/df.std(ddof=0)

#load the data
df = pd.read_csv('/Users/rosskearney/Desktop/Fin. Data Sci./Tutorials/HomeWork8/creditcard.csv')

df['Class'].sum()

# df = df[(np.abs(stats.zscore(df)) < 3).all(axis=1)]

# split the dataset in training and test datasets
train, test = train_test_split(df, test_size=0.3, shuffle=False)

cols = ['V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10', 'V11', 'V12', 'V13', 'V14' \
        , 'V15', 'V16', 'V17', 'V18', 'V19', 'V20', 'V21', 'V22', 'V23', 'V24', 'V25', 'V26' \
        , 'V27', 'V28']

xtrain = train[cols]
ytrain = train['Class']
xtest = test[cols]
ytest = test['Class']

# =====
#           Synthetic Minority Oversampling Technique
# =====

smote = SMOTE(random_state = 101)
xtrain_oversample, ytrain_oversample = smote.fit_resample(xtrain,ytrain)
xtest_oversample, ytest_oversample = smote.fit_resample(xtest,ytest)

XGB = XGBClassifier(n_estimators=30, n_jobs=-1, verbose=1, use_label_encoder=False)
XGB.fit(xtrain, ytrain, eval_metric=['aucpr'], eval_set=[(xtrain_oversample, ytrain_oversample)], (xtest_oversample, ytest_oversample))

# Plot feature importance
plot_importance(XGB)
plt.show()
predclasstrain = XGB.predict_proba(xtrain_oversample)[:,-1]
predclasstest = XGB.predict_proba(xtest_oversample)[:,-1]
print('Training PR AUC:' + "{:.3f}".format(average_precision_score(ytrain_oversample,predclasstrain)))
print('Test PR AUC:' + "{:.3f}".format(average_precision_score(ytest_oversample,predclasstest)))
```

```
# original  
# Training PR AUC:0.985  
# Test PR AUC:0.779
```
