ECON 425 HW5 Solutions

February 7, 2024

Problem 1 (precision-recall tradeoff)

Suppose that you fit a binary classification model that yields accuracy of 80% on a test set with an equal number of positively and negatively labeled observations.

(i) Express precision and recall as functions of the overall proportion of true positives $\gamma = TP/N$, where TP is the number of true positives and N is the test sample size. What is the minimal/maximal value of precision that can be achieved when you vary γ? How about recall? On a single graph, draw precision and recall as functions of γ.

(ii) Express precision as a function of recall and draw its plot.

Solution

(i) The confusion matrix is

$$\begin{pmatrix} \gamma N & \gamma N - 0.3N \\ 0.5N - \gamma N & 0.8N - \gamma N \end{pmatrix}.$$

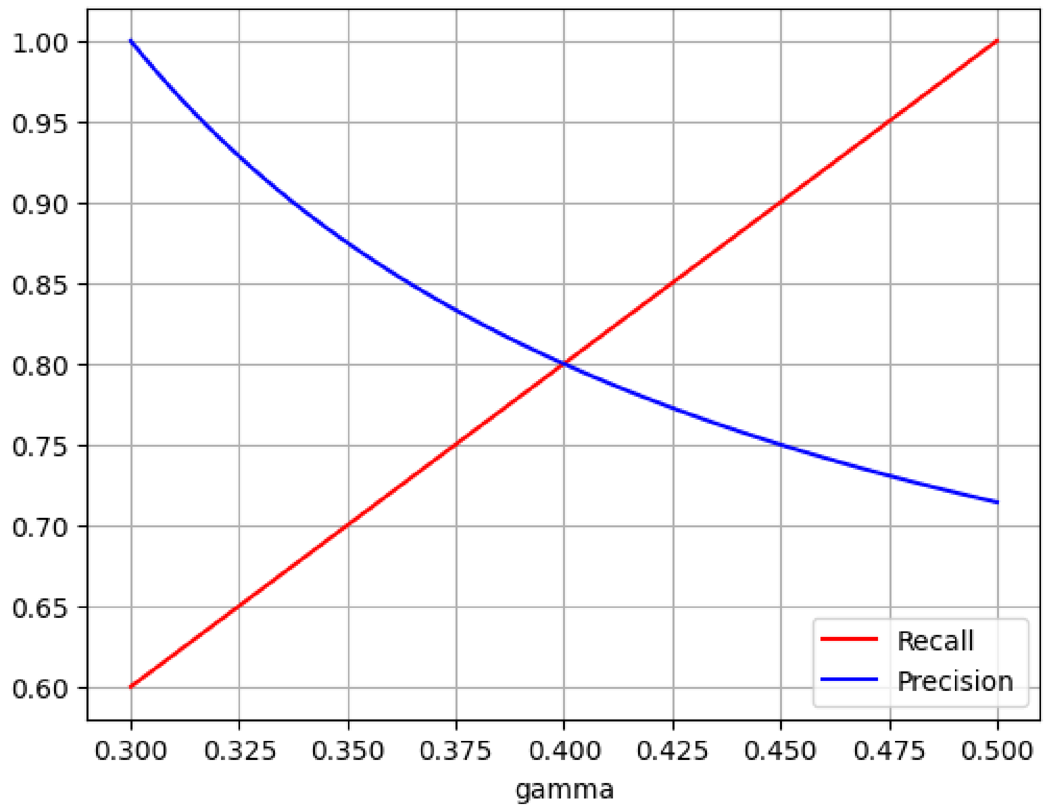
Therefore,

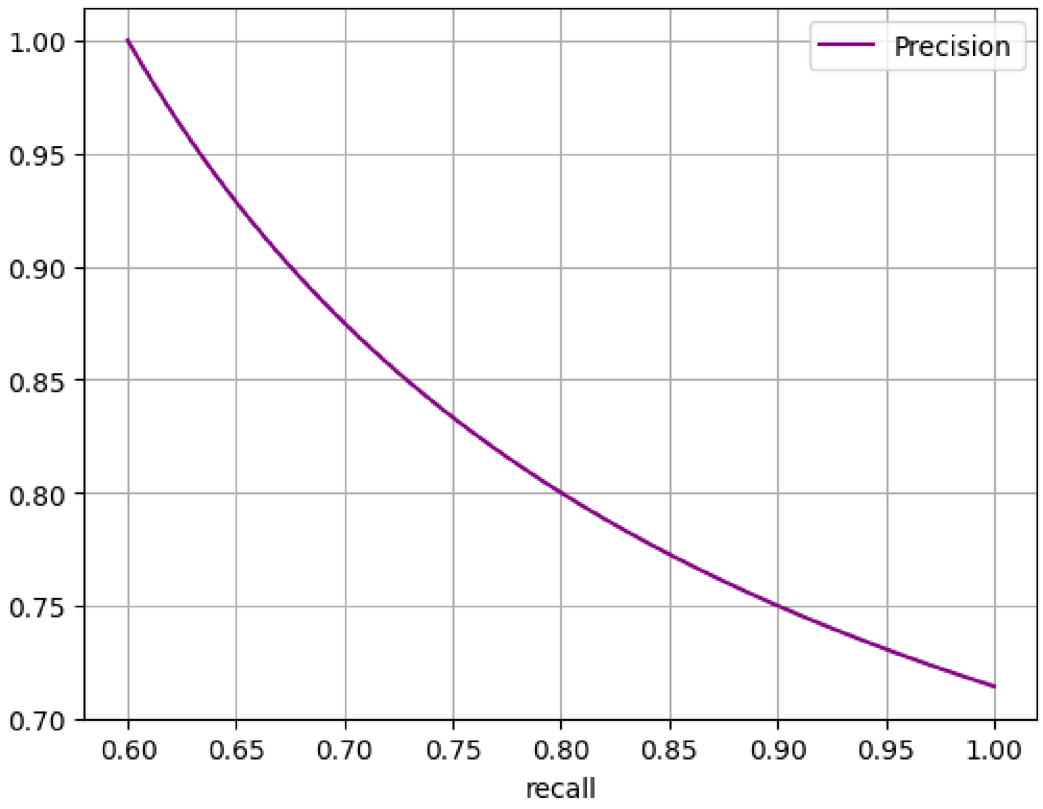
$$\text{Recall} = \frac{\gamma N}{0.5N} = 2\gamma,$$
$$\text{Precision} = \frac{\gamma N}{2\gamma N - 0.3N} = \frac{\gamma}{2\gamma - 0.3}.$$

(ii) Multiplying the numerator and the denominator of Precision by 2, we obtain

$$\text{Precision} = \frac{2\gamma}{2 \cdot 2\gamma - 0.6} = \frac{\text{Recall}}{2 \cdot \text{Recall} - 0.6}.$$

[27]:
```python
# PLOTS FOR PROBLEM 1

import matplotlib.pyplot as plt
import numpy as np

# Plot for (i)
rec = lambda gamma: 2*gamma
prec = lambda gamma: gamma/(2*gamma-0.3)
gamma_range = np.linspace(0.3, 0.5, 100)
plt.figure()
plt.plot(gamma_range, rec(gamma_range), color='red')
plt.plot(gamma_range, prec(gamma_range), color='blue')
plt.grid()
plt.legend(['Recall', 'Precision'])
plt.xlabel('gamma')
plt.show()

# Plot for (ii)
prec_rec = lambda rec: rec/(2*rec-0.6)
rec_range = np.linspace(0.6, 1, 100)
plt.figure()
plt.plot(rec_range, prec_rec(rec_range), color='purple')
plt.grid()
plt.legend(['Precision'])
plt.xlabel('recall')
plt.show()
```





Problem 2 (SMOTE-NC)

The file card_transdata.csv contains data on 1,000,000 credit card transactions. The goal is to fit and test a model predicting whether a given transaction is fraudulent. Let the first 500,000 observations be the training set and the remaining observations be the test set.

Since the dataset contains both continuous and categorical features, while SMOTE can only handle continuous features, we use SMOTE-NC, a version of SMOTE that can handle both. In Python, use imblearn.over_sampling.SMOTENC. Do not forget to provide information about the categorical features to SMOTENC.

(i) For each $\gamma \in \{0.1, 0.2, \ldots, 1\}$, oversample the training set with the desired ratio γ of the number of samples in the minority class over the number of samples in the majority class.

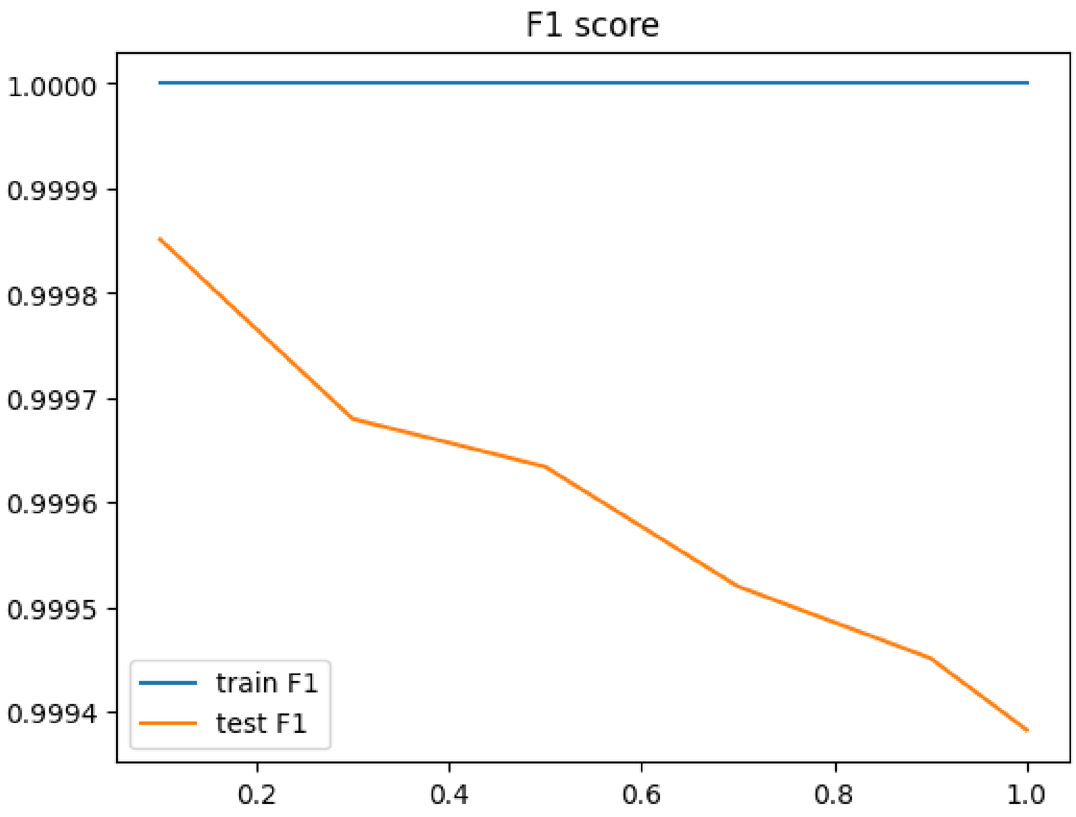
(ii) For each of the resulting oversampled datasets, fit a decision tree (with information gain as the feature selection criterion) and calculate its training and testing recall, precision, and F_1 score.

(iii) Draw a plot of training and testing F_1 score as functions of γ. Does using SMOTE-NC improve the decision tree's performance?

[28]:
```
import pandas as pd
import numpy as np
from imblearn.over_sampling import SMOTENC
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import f1_score, precision_score, recall_score

datapath = '/Users/franguri/Library/CloudStorage/Dropbox/_TEACHING_/Econ 425 ML␣
↪UCLA Winter 2024/WEEK 5 Imbalanced data/card_transdata.csv'
data = pd.read_csv(datapath, header=0)

n_train = 500000
x_all = data.drop('fraud', axis=1)
y_all = data['fraud']
x_train, y_train = x_all[:n_train], y_all[:n_train]
x_test, y_test = x_all[n_train:], y_all[n_train:]

print('Share of fraudulent transactions = ', np.mean(y_all))
print('Share of fraudulent transactions train = ', np.mean(y_train))
print('Share of fraudulent transactions test = ', np.mean(y_test))

min_to_maj_range = [0.1,0.3,0.5,0.7,0.9,1]
train_F1 = np.empty((len(min_to_maj_range),))
test_F1 = np.empty((len(min_to_maj_range),))
train_recall = np.empty((len(min_to_maj_range),))
test_recall = np.empty((len(min_to_maj_range),))
train_precision = np.empty((len(min_to_maj_range),))
test_precision = np.empty((len(min_to_maj_range),))

for i in range(len(min_to_maj_range)):
    sm = SMOTENC(categorical_features=['repeat_retailer', 'used_chip',␣
↪'used_pin_number', 'online_order'],
                 random_state=0, k_neighbors=5,␣
↪sampling_strategy=min_to_maj_range[i])

    x_res, y_res = sm.fit_resample(x_train, y_train)
    print('>>>>>>>> min_to_maj =', min_to_maj_range[i])
    print('resampled data: n_obs =', y_res.shape[0], 'share of fraud =', np.
↪mean(y_res))

    # DECISION TREE
    clf = DecisionTreeClassifier(criterion='entropy', random_state=0)
    clf.fit(x_res, y_res)
    train_F1[i] = f1_score(y_res, clf.predict(x_res))
    print('Train F1 =', train_F1[i])
    test_F1[i] = f1_score(y_test, clf.predict(x_test))
    print('Test F1 =', test_F1[i])
    train_precision[i] = precision_score(y_res, clf.predict(x_res))
    train_precision[i] = precision_score(y_test, clf.predict(x_test))
    train_recall[i] = recall_score(y_res, clf.predict(x_res))
    train_recall[i] = recall_score(y_test, clf.predict(x_test))
```

```python
plt.figure()
plt.plot(min_to_maj_range, train_F1)
plt.plot(min_to_maj_range, test_F1)
plt.title('F1 score')
plt.legend(['train F1', 'test F1'])
plt.show()
```

```
Share of fraudulent transactions =  0.087403
Share of fraudulent transactions train =  0.087334
Share of fraudulent transactions test =  0.087472
>>>>>>>> min_to_maj = 0.1
resampled data: n_obs = 501966 share of fraud = 0.09090854759087269
Train F1 = 1.0
Test F1 = 0.9998513589224665
>>>>>>>> min_to_maj = 0.3
resampled data: n_obs = 593232 share of fraud = 0.2307680637592038
Train F1 = 1.0
Test F1 = 0.9996799122044904
>>>>>>>> min_to_maj = 0.5
resampled data: n_obs = 684499 share of fraud = 0.33333284635916194
Train F1 = 1.0
Test F1 = 0.999634202103338
>>>>>>>> min_to_maj = 0.7
resampled data: n_obs = 775766 share of fraud = 0.41176463005597047
Train F1 = 1.0
Test F1 = 0.9995199451365872
>>>>>>>> min_to_maj = 0.9
resampled data: n_obs = 867032 share of fraud = 0.4736837856042222
Train F1 = 1.0
Test F1 = 0.9994513909525226
>>>>>>>> min_to_maj = 1
resampled data: n_obs = 912666 share of fraud = 0.5
Train F1 = 1.0
Test F1 = 0.9993828853538124
```


F1 score
1.0000
0.9999
0.9998
0.9997
0.9996
0.9995
0.9994
0.2
0.4
0.6
0.8
1.0
train F1
test F1